

SAS 在财务研究中的应用

林煜恩 著

電子工業出版社

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书分为两部分，第一部分为第 1 章到第 7 章，主要针对财务管理、金融、投资领域需要用到的 SAS 语法进行介绍；第二部分为第 8 章到第 17 章，主要针对财务实践研究中会使用的方法进行介绍。

本书的第二部分中提供了标准化的宏语法，在第 8 章中提供了 stat 以及 correlation 两个宏语法，读者仅需将文档准备好，输入文档名称，希望进位到小数点后几位以及要分析的变项名称，宏语法就能够直接将叙述统计表以及相关系数表自动生成出来，第 9 章提供了 npar 及 ttest 两个宏语法，可以快速地进行两群体的中位数及均值检定，第 10 章提供了 Jagaseesh and Titman 动能投资策略的语法，而第 11 章到第 15 章阐述了各个回归模型，以及相对应的宏语言，提供给读者进行实践分析，第 16 章则针对短期及长期事件研究法做了详细的介绍，最后在第 17 章则介绍了如何使用财务数据构建出 Fama and French 的 5 因子报酬率。

本书适用于高等院校财经类专业本科生以及硕士研究生，它提供了撰写学术论文相当好用的宏语法；而对于博士研究生而言，本书撰写的宏语法逻辑，也可以提供其在进行更进一步的模型分析时的撰写架构。另外，本书对其他行业从事数据分析的工作人员也有一定的参考价值。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目 (CIP) 数据

SAS 在财务研究中的应用 / 林煜恩著. —北京：电子工业出版社，2017.1

ISBN 978-7-121-30225-1

I. ①S… II. ①林… III. ①统计分析—应用软件—应用—财务管理 IV. ①C819 ②F275-39

中国版本图书馆 CIP 数据核字 (2016) 第 258355 号

策划编辑：师纬风

责任编辑：徐津平

特约编辑：顾慧芳

印 刷：

装 订：

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本：787×980 1/16 印张：23 字数：510 千字

版 次：2017 年 1 月第 1 版

印 次：2017 年 1 月第 1 次印刷

印 数：3000 册 定价：69.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888, 88258888。

质量投诉请发邮件至 zltz@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819 faq@phei.com.cn。

推荐序一

能得天下英才而教之是身为老师最感到快乐的事情，这也是十多年来，我的学生林煜恩给我的感受。我是他硕士班以及博士班的指导老师，直到他取得博士学位之后，我们仍持续进行学术研究。如今得知煜恩要出书了，身为老师的我真的为他感到高兴。

煜恩在硕士班的时候就崭露了对于撰写 SAS 程序的天赋与热诚。让我印象最深刻的是他曾经为了写出好的程序，连续花了六个小时的时间待在计算机屏幕前仔细看着程序运行中的 LOG 文件，并藉此观察出 SAS 速度越变越慢的原因，逐步改进，这样的努力让他从一个不会写程序的门外汉变成一个热爱写程序、并能自行写程序来解决问题的专家。现在的煜恩不仅擅长 SAS 指令的撰写，亦将其这些年来的 SAS 撰写经验编纂成书，他将 SAS 程序以简驭繁，将过去复杂的研究语法变得简洁方便，且可以让 SAS 初学者能快速上手。

事实上这本书最早是煜恩为了让学弟学妹可以更容易使用 SAS 软件，而把自己研究 SAS 的心得写成的秘籍。还记得煜恩将本书第一版的书名取为《恶魔党私传秘籍》。那时我看到这本秘籍，真的很开心，因为这是一本学长对学弟学妹的关爱，也是煜恩为师门留下的宝贵资产。这几年来，我指导的学生们（也是煜恩的学弟学妹们）在研究时遭遇到的问题，煜恩也不吝给予帮助，同时也修正并更新了他的秘籍。

煜恩曾经开授过财务软件应用（SAS）的课程，他的教学态度认真积极，与学生相处融洽，学生们非常喜欢向他请教 SAS 问题。许多学生第一次跑程序，对于程序往往很彷徨，煜恩非常了解初学者常犯的错误及不懂之处，因此他所撰写的秘籍可以让初学者更简单地认识 SAS，不再害怕写程序，深获学生们的好评。

在这样的情况下，经过煜恩不断的研究、修改与更新，这本秘籍逐渐形成为各位读者眼前的这本书，字里行间所透露的，是煜恩对后进入者的叮咛与指导，也是他送给我的最佳礼物。

我在此向希望踏入学术界的初学者们强烈推荐《SAS 在财务研究中的应用》这本书。这本书由浅入深，一步一步带领着读者进入 SAS 这个领域，指引着读者在论文实证上使用更好、更有帮助的程序代码。本书清楚地写出了每个程序代码的含义以及使用上常犯的错误，并搭配着程序跑出的结果，让读者更容易看懂程序语言。若能跟着作者引导的步骤一步一步地尝试，相信你们很快就会使用 SAS，甚至可以成为 SAS 达人，使你们的研究更为顺利。在此，我也强烈建议已经踏入学术界已有写程序经验、想另外学习 SAS 的学者们将此书作为一本工具书，当你所写的程序跑得太慢或者遇到瓶颈时，相信这本书会提供许多的帮助。

池祥萱

台湾东华大学财务金融学系教授

推荐序二

撰写论文主要是在学术理论及实证研究中的新发现，现今学术领域研究的过程中，由于统计软件、绘图软件等工具可使用，使得学术领域发展更为迅速，产出更多。

撰写一篇好的论文，除了有好的想法之外，还需要搜集整理资料、实证研究及稳健性检验。研究发展刚开始时，资料搜集较为困难，且资料量较少；到现今有许多数据库提供给研究人员使用，数据收集简单许多但数据量越来越庞大，使得数据的整理及合并数据库较为困难，因此需要有好的软件协助。当数据整理完成后，紧接着实证研究的部分，需选择适当的统计模型且正确地计算出结果，此时依然需要好的软件功能才能快速且正确地完成。而 SAS 在财务数据处理的功能上，是其他统计软件无法比拟的。但当前教育课程的设计，财务专业的学生虽然具备足够的财务知识，可是对各种财务软件的特性却是一知半解，更缺乏处理数据的能力；而当前有关 SAS 软件的书籍，多是针对生物统计、市场营销领域撰写的，鲜少有专门为财务专业背景而撰写的 SAS 教材；而针对财务专业的 SAS 教材中，其程序过于偏向传统指令的介绍，对于进阶编程技巧较少涉猎，导致运行效能较差，应用价值不高。

我很高兴得知煜恩针对财务专业撰写这本教科书，它突破过去教材传统思维，提供了不同的视角，确实与应用紧密结合。这本教科书可成功指引本科生以及研究生在论文实证上使用更好且更正确的程序语言，并且搭配 SAS 所提供的功能，将软件使用得淋漓尽致。本书清楚地写出了财务上经常使用的 SAS 程序语言，并且介绍各种财务模型及投资组合策略的程序语言，这样的分析手法使程序语言更为清楚，并且提供许多统计模型及宏语法，可以快速地整理出论文所需要的表格，也有许多细节部分也说明得非常仔细，让初学者不再害怕程序语言，使资深的财务人员节省许多时间，我相信读者熟读这本书之后，可以感觉到作者的用心。

在这本《SAS 在财务研究中的应用》中，煜恩从许多不同的角度，分析程序语言中过去没有想过的方式。通过他的诠释，读者会发现许多复杂的程序语言，变得简单而且容易了解，同时也凸显了各种统计模型以及形成投资组合的差异。读者往往会通过那些投资组合 SAS 之模型，验证自己所学的理论模型。更难得的是，借助这本教科书，读者可非常简易地通过参数的修改，进行更多不同的研究或形成不同的投资组合，事半功倍。相信经过本教科书的介绍，读者可以自行发挥，通过自我学习的过程，获得更多、更广泛的专业知识。

蕭 朝 興

台湾东华大学财务金融学系教授

Tel: +886-38633135

Fax: +886-8633130

Email: cschiao@mail.ndhu.edu.tw

前言

12 年前，笔者于台湾东华大学攻读企业管理系硕士班，当时完全没有编程基础就必须学习动能投资策略（见本书 10.3 节与 10.4 节）的语法，所以就一股脑儿一行一行运行程序，一行一行修改，在暑假期间，每天盯着电脑的日志文件六个钟头，与同行见面讲的话就是，“你的程序跑得如何？”当时觉得研究财务金融的人好苦，为了做好财务金融研究，不得不去接触从大学到研究生前期都不会接触的统计软件，因为是研究生，所以这些东西要自己学；因为要完成论文，这些东西拼死拼活也得学好，也因此开始了我学 SAS 的路。

10 年前开始攻读博士学位后，要引领硕士生学习 SAS，也因此要接触更多的研究方法，更多的研究语言，当时很多语法都是 20 世纪 90 年代的前人留下来的语法，程序确实能够顺利运行，但是问题是运行速度慢。跟我合作的老师问，“煜恩，为什么程序运行得这么慢，却没有人要去改善呢？”，当时我没有答案，但当年我指导一名硕士生，请他把程序改好，使运行结果会比较快时，他这么回答：“学长，我没有时间去改程序，我还是会把结果跑出来，自己再动手转到 Word 感觉会比较快。”于是我得到答案了，大家都知道运行慢，可是没人有勇气去修改这些程序，因为不知道会花多少时间才能修改完，而就算完成以后速度变快，自己也毕业了，以后也用不上了，所以如何能期待硕士生修改这些语法呢？

于是我开始撰写一本讲义，专门为我师门“恶魔党”写的 SAS 攻略，最早的书名为“SAS 在财务研究上的应用：恶魔党的私传秘籍”。最初的版本仅有 129 页，内容上以财务领域常常涉及的语言为主，之后接触到 SQL 语言以及 SAS 中的 Output Delivery System（ODS）的功能，更进一步开启了笔者将实践表格输出到 Excel 的进程，于是在本书的第 9 章到第 17 章所呈现的 SAS 语法，都能将实验结果整理成可发表的论文格式，并且将其输出到 Excel 文档，可以节省人工输入结果的时间，我想如果这本书可以帮助读者节省进行实验的时间，不管是学生、研究者都能够把时间花在更有生产力的工作上。

虽然本书主要是针对 SAS 在财务研究上的编程，但是 SAS 在应用上不仅限于财务研究分析，SAS 除了是一款优良的统计软件外，还是一家跨国公司的名称，其主要是为公司以大数据的思维逻辑构建适合的数据库，并且提供良好的分析模组，其运用层面包含商业智能分析、客户智能、数据管理、决策管理器、绩效管理、风险管理、供应链管理，而由于 SAS 本身最早起源于大型数据库的分析，因此在大数据背景下，还具备了云分析以及文本分析等功能，对于各行各业都起着重要的功用，其目前应用的行业包含教育、医疗保健、制造业、媒体分析、旅游与运输业、汽车行业、电信业、零售业、资本市场、银行业以及保险业等相关行业。在美国，如果取得了 SAS 的 Base 以及 Advance 的国际证照，一般都能取得 10 万美元以上的年薪。

而在大数据时代，金融行业面临着微信钱包、余额宝等手机理财侵蚀传统银行业提供的投资理财

服务，又面临着 P2P 网贷来抢占传统消费金融贷款以及企业金融贷款的业务，在此情况下，金融业将由传统以人力为主的服务转向数据挖掘的服务，在理财上，需要了解什么样的客户会希望接受人力服务的理财规划，或者是提供相关的理财材料供其选择；在借贷上，需要快速地了解该位客户未来违约的可能性，这些都需要进行数据收集规划，并且进行分析，而 SAS 公司这几年来还为大型银行提供其数据挖掘以及消费者信用分析的定制化服务，在全球 100 强银行中，有 99 家银行使用 SAS 的产品服务，应运而生的是，这 99 家银行产生了对于专精 SAS 的金融从业人员的需求，而这样的趋势现象也在我国逐渐蔓延。

在我国，SAS 自从 2013 年开始举办汇丰杯中国高校数据分析大赛以来，除了提供高额奖项外，也提供了企业招才的机会，以 2015 年的赛事为例，SAS 还邀请了毕马威、京东金融、中信银行、丰田汽车金融（中国）有限公司、交通银行太平洋信用卡中心、南方航空电子商务部、广汽汇理汽车金融有限公司、奇瑞徽银汽车金融股份有限公司、招商银行、平安银行、京东方、光大银行等企业作为大赛的颁奖者，且这些企业也提供参赛同学实习以及未来工作的机会，这说明中国金融行业已经开始运用 SAS 作为进行数据分析的主要工具，而 SAS 更是未来金融业学生必备的工作技能。

最后，在进行数据分析的过程中，我常提醒自己，“如果数据站在你这边，就引用数据；如果理论站在你这边，就引用理论；如果两者都背离你而去，你不是发现了新的现象，就是做错了，而放弃是在做错的时候才做的决定。”也希望各位读者，在进行研究分析时，共勉之。

林煜恩

2016 年于吉林大学匡亚明楼

目 录

第 1 章	SAS 入门介绍	1
1.1	SAS 的基本接口介绍	2
1.2	SAS 语法的基础架构	2
1.3	如何输入数据	4
1.4	如何输出数据	13
1.5	总结	15
第 2 章	SAS 数据的运算与函数	16
2.1	四则运算	17
2.2	统计函数	18
2.3	随机函数	21
2.4	时间函数	22
2.5	文本变量的处理	26
2.6	总结	30
第 3 章	数据与变量的产生和选取	31
3.1	利用 SAS 产生数据	32
3.2	保留、删除变量	36
3.3	保留、删除观测值	38
3.4	抽样方法	38
3.5	总结	43
第 4 章	数据的排序、分组与转置	44
4.1	数据的排序 (proc sort)	45
4.2	数据的分组 (proc rank)	49
4.3	数据的转置	53
4.4	总结	57

第 5 章 数据的合并	58
5.1 垂直合并	60
5.2 水平合并	65
5.3 总结	72
第 6 章 SAS 的数据库管理	73
6.1 文档的复制、删除与保留 (proc datasets)	74
6.2 结构化查询语言	78
6.3 总结	91
第 7 章 宏语法 (%macro)	92
7.1 基础宏语法	93
7.2 进阶宏语法	96
7.3 宏语法撰写技巧	107
7.4 总结	111
第 8 章 描述统计	112
8.1 常见的描述统计量	113
8.2 相关系数	121
8.3 个人化表格宏解析	126
8.4 趋势图基础语法介绍	129
8.5 离群值的处理 (winsorize)	140
8.6 总结	144
第 9 章 两群体差异性检定	145
9.1 均值检定	146
9.2 中位数检定	151
9.3 两群体检定宏进阶用法	154
9.4 总结	156
第 10 章 投资组合与报酬率检定	157
10.1 投资组合股票的数目与风险	158
10.2 效率前缘的绘制	164
10.3 初探动能投资策略	168
10.4 再探动能投资策略	177
10.5 Newey and West 的调整语法	185
10.6 总结	190

第 11 章 基础回归语法	191
11.1 回归语法介绍	192
11.2 格式化回归模型输出	201
11.3 Fama-MacBeth 回归模型	212
11.4 总结	222
第 12 章 回归语法的应用	223
12.1 移动窗口 (moving window)	224
12.2 共同基金绩效评估: 移动窗口的应用	229
12.3 滚动法 (Rolling)	234
12.4 Where 语法有妙招	239
12.5 结构性改变	241
12.6 分段回归 (piecewise regression)	246
12.7 总结	254
第 13 章 panel data (proc panel、proc tscsreg)	255
13.1 固定效应与随机效应的估计方法	256
13.2 panel data 的实证流程	261
13.3 格式化 panel data 模型输出	264
13.4 总结	267
第 14 章 罗吉斯特模型	268
14.1 logit model (logistic regression)	269
14.2 conditional logistic regression	276
14.3 multinomial logistic regression	280
14.4 分类与概率转换	287
14.5 总结	291
第 15 章 tobit 模型 (proc lifereg)	292
15.1 受限数据 (censored data) 与截断数据 (truncated data)	293
15.2 格式化 tobit 模型输出	300
15.3 总结	302
第 16 章 事件研究法	303
16.1 短期事件研究法	304
16.2 日历期间投资组合法	312
16.3 买进持有异常报酬率: 配对投资组合法	318

16.4	买进持有异常报酬率：配对样本法	326
16.5	总结	335
第 17 章	特殊议题	336
17.1	均值抽样分配	337
17.2	拔靴法 (Bootstrap method)	339
17.3	构建 5 因子与动能因子报酬率	347
17.4	总结	356

第 1 章

SAS 入门介绍

对于大多数人而言，SAS 似乎是个很难入门的软件，因为它需要自行撰写程序。但是这也意味着 SAS 的自由度相当高，用户可以自由地撰写程序。本书采用的是 SAS 9.2 版本，书中的部分语法只兼容于 SAS 9.2 以上的系统；而 Proc Panel 语法是从 SAS 9.2 以后开始提供的，若使用 SAS 9.2 以前的版本，可能无法使用。

首先简单介绍一下如何使用 SAS。SAS 由三大部分组成：editor、log 和 output，如表 1-1 所示。

表 1-1

editor	SAS 的程序编辑器，所有的程序皆在此输入
log	记录执行 SAS 过程中遇到的问题。如果 SAS 发生问题，常会出现绿色的警告 (warning) 和红色的错误 (error)，初学者往往会因此以为程序写错。事实上有些警告和错误是在撰写通用程序时产生的，这些问题不会影响 SAS 结果的正确性。不过真正严重的问题是 SAS 中止程序运行，这时就需要进行调试 (debug)
output	SAS 执行程序后的输出结果

接下来将逐一介绍以下内容：SAS 的相关基本接口；SAS 语法的基础架构；如何读取外部文件，以应用在其后的数据处理和一些程序上，文件分为记事本数据、Excel 文件数据及外部的 SAS table 文件数据；如何将已经整理好的 table 文件输出到外部文件，以使用户使用不同的统计软件进行数据的相关验证。

1.1 SAS 的基本接口介绍

初始进行程序的编写与执行时，需要善用工具栏中的相关工具，如图 1-1 所示，相关操作说明如表 1-2 所示。

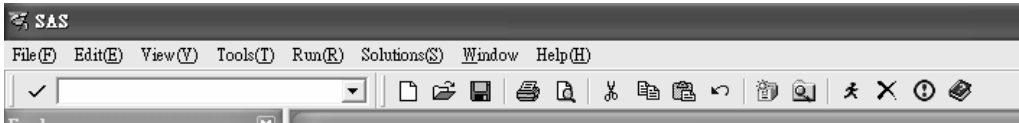


图 1-1

表 1-2

	这三个图标是常见的文件管理图示，分别为打开新文件、打开旧文件和存储文件
	程序的打印图标，便于用户教学调试等。一开始不熟悉程序时，可以将程序打印出来，避免长期观看计算机屏幕
	在编辑程序时，有时候会用到相同的语法，这时就可以使用剪切、复制、粘贴操作来处理数据。如果有错也可以撤销
	手动输入新的逻辑库和检验 SAS 逻辑库，后者较常用
	程序的执行（submit）、清除所有程序、结束程序和 SAS 的 help 选项（查询指令）。对初学者最重要的是小黑人图标和惊叹号图标。进阶使用者要常常使用 help 选项来进行查询

在进行 SAS 程序的撰写时，有时候还要使用 Edit (E) 中的 Find 和 Replace。显而易见，SAS 程序的编辑方法与 Word 文档编辑文字的操作非常相似。在 SAS 程序编辑中单击鼠标的右键，也可以看到与 Word 界面相似的复制、粘贴等相关编辑功能，这使得我们在进行 SAS 的程序编辑时，可以很快地适应相关操作。接下来逐步介绍如何编辑 SAS 程序。

1.2 SAS 语法的基础架构

在 SAS 语法中总共有两大体系，第一体系为 data step (数据步)，第二体系为 proc step (程序步)。

在 data step 中，主要是进行数据与变量的整理。财务领域内公司的数据类型非常多，有股票交易数据、公司财务报表数据，甚至还有多年期的数据，因此需要对不同数据源的数据进行整理，这些整理步骤往往是在 data step 中进行整理的。除了整理数据源外，变量的整理也是财务数据整理中的一大重头戏，该部分可以分为两类：一类为变量的横向处理，即所有的数据都是同一公司、同一个年度的数据，这部分通过四则运算便可完成；另一类是变量的纵向处理，在财务研究中，往往要使用公司前一年度的数据，即会使用滞后的变量，如计算营收增长率。在数据库中，所有公司的数据是按顺序排列的，编程时就需要注意滞后的变量是否会取到另外一家公司的值。

在 proc step 中，就是使用 SAS 设计好的规范程序。笔者根据个人的财务实证经验，将 SAS 的程序步划分为三类：变量类、模型类和数据库类。变量类，顾名思义，是针对变量进行排序、分组、运算等功能的程序，在撰写过程中通常带有 var (variable; SAS 通常是采用英文单词的前三位)，其结尾都以 “run;” (运行) 结束。模型类是进行实证时的语法，撰写过程中都带有 model，语法结尾多缀以 “quit;” (停止) 结束。数据库类的程序通常不以单一数据中的变量作为整理对象，其运行多以整个或多个数据文件为对象。在撰写程序过程中常常会出现过渡数据，这不是我们最终要保留的数据，此时可以利用数据库类的程序步将其删除。以下为笔者进行实证分析时常用到的程序步¹，如表 1-3 所示。

表 1-3

变量类	
程序步	程序说明
proc sort	可将数据依照指定的变量进行排序，是非常常用的程序步
proc rank	可将数据依照指定的变量进行分组，是投资组合分析中常用的程序步
proc transpose	可将数据依照指定的变量进行分组
proc means	可针对指定的变量进行描述统计，与 proc univariate 相似
proc univariate	可针对指定的变量进行描述统计，与 proc means 相似
proc corr	可针对指定的变量进行相关分析
模型类	
程序步	程序说明
proc reg	进行普通最小二乘法 (OLS) 的回归程序步
proc model	可撰写各种模式，本书籍以计算 Newey West 的估计
proc panel	进行面板数据的程序步
proc logistic	进行罗吉斯特的程序步
proc probit	进行波比特的程序步，编程语句逻辑和 proc logistic 类似
proc lifereg	本书主要用来进行 Tobit model 的程序
proc mixed	混合线性模式，可运行随机系数模式
proc arima	进行时间序列中的自回归以及移动平均的模式
数据库类	
程序步	程序说明
proc contents	可检验数据的各项内容，确认数据的各项特征、长度
proc datasets	针对数据数据进行处理
proc copy	可将目标的文件复制到逻辑库

1 这种分类方法仅是笔者个人进行实证分析的心得分类，并非是规定的分类。

续表

proc delete	可将目标的文件删除
proc append	可将新文件附加到新文件，特别注意变量的长度尽量要一致
proc sql	结构化查询语言，是运用性相当重要的数据库语言
proc compare	比较两个文件的程序步
proc import	将外部数据导入 SAS，通常是导入 Excel 以及 CSV 文件
proc export	将 SAS 数据导出到外部文件，通常是导出 Excel 以及 CSV 文件
proc surveyselect	随机抽样语法，针对文件抽取数据，进行拔靴复制法的重要程序步

笔者根据整理财务数据的经验，认为数据处理大多由四部分构成：第一部分是数据的水平处理，第二部分是数据的垂直处理，第三部分是数据的转置处理，最后一部分是数据的配对处理。前两个部分使用数据步进行处理的，后两个部分采用程序步处理，其中转置处理使用 proc transpose，配对处理则采用 proc sql。只要灵活掌握这四部分的处理逻辑，就可以轻松地将所有数据整理完毕。

1.3 如何输入数据

SAS 本身可以读取外部文件，如记事本数据和 Excel (CSV) 文件的数据。一般而言，Excel 可以采用 SAS 本身的 import 操作输入数据，所以很容易上手，下面介绍具体操作。

1. 在程序中输入数据

输入数据的简单程序示例：

```
data a;  
input id a $ b c;  
datalines;  
1101 a 2 3  
1102 b 3 4  
1103 d 2 4  
;  
run;
```

该语法会生成一个 SAS 的 table，命名为 a，并输入 id、a、b、c 四个变量，datalines 的作用是告知 SAS 下面要输入数据，接着为四种数据，结束时输入 run，SAS 就会将数据存到表 a 中，如图 1-2 所示。

	id	a	b	c
1	1101	a	2	3
2	1102	b	3	4
3	1103	d	2	4

图 1-2

在上例中，语法结束时，都会以分号 “;” 来区分。这里要注意一点，如果是在程序中输入数据，分号不能与数据在同一行。

```
data a;  
input id a $ b c;  
datalines;  
1101 a 2 3  
1102 b 3 4  
1103 d 2 4;  
run;
```

如图 1-3 所示为出现程序警告的数据输入实例，第三笔数据无法读进去，检验 SAS 的 log 文件，发现对该语法提出了警告。ERROR 中的信息显示该语法无效或者顺序不适当。

在此提醒一点，如果要在程序中输入数据，分号必须在数据列的下一行独立存在。图 1-3 显示的例子中数据都是以空格隔开的，所以导致第三笔数据无法读进去。如果数据本身是链接在一起的，可以按照下列方法来读取。

```
data a;  
input id 1-4 a $ 5 b 6 c 7;  
datalines;  
1101a23  
1102b34  
1103d24  
;  
run;
```

首先，id 1-4 指的是读取以下数据 1 到 4 栏为 id 变量，a 为第 5 栏的数据，前面有美元符号（\$）表示其为文本类变量，第 6 栏读取为 b，第 7 栏读取为 c。这样的读取方式在读取记事本数据（txt 文件）时也可以适用。

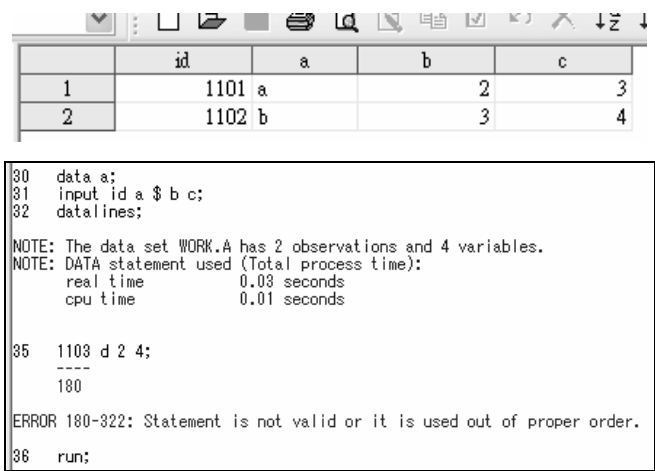


图 1-3

有时候数据形式如时间 1996 年 1 月,要想将其拆成年、月的显示格式(笔者个人偏好的读取方式),就需要读取 1-4 这样的字段数据。但如果数据本身使用空格键进行区隔,例如 1 月之后紧接着空白数据,那么此时只要输入 m、id、a 等,SAS 就会自动区分读取数据了。具体程序如下,生成的对应表格如图 1-4 所示。

```
data a;
input y 1-4 m id a $ b c ;
datalines;
199601 1101 a 2 3
199602 1102 b 3 4
199603 1103 d 2 4
;
run;
```

	y	m	id	a	b	c
1	1996	1	1101	a	2	3
2	1996	2	1102	b	3	4
3	1996	3	1103	d	2	4

图 1-4

在撰写程序时,可能会觉得将月数据分拆成两个变量不适当,于是加入 month 1-6 的命令,希望 SAS 重复读取 month。以数据 199601 为例,我们可以读出三个变量 1996、1 和 199601。这里需要注意 SAS 有一个特性,当指定字段后,下一个变量若不指定字段,数据就会从上一个指定字段的地方重新读取。该部分的程序如下所示,生成的对应表格如图 1-5 所示。

```
data a;
input y 1-4 m id month 1-6 a $ b c ;
datalines;
199601 1101 a 2 3
199602 1102 b 3 4
199603 1103 d 2 4
;
run;
```

	y	m	id	month	a	b	c
1	1996	1	1101	199601	1101	.	2
2	1996	2	1102	199602	1102	.	3
3	1996	3	1103	199603	1103	.	2

图 1-5

由图 1-5 可以了解到 SAS 在读取数据 199601 之后,a 变量又重新读取 1101 这笔数据。因此如果要重复读取数据,只有下列两种读法。

第一种方法是在读取完所有数据之后,另外要求 SAS 读取第 1~6 栏的数据。具体程序如下,生

成的对应表格如图 1-6 所示。

```
data a;
input y 1-4 m id a $ b c month 1-6;
datalines;
199601 1101 a 2 3
199602 1102 b 3 4
199603 1103 d 2 4
;
```

	y	m	id	a	b	c	month
1	1996	1	1101	a	2	3	199601
2	1996	2	1102	b	3	4	199602
3	1996	3	1103	d	2	4	199603

图 1-6

第二种方法是在换字段读取之前，立刻读取该字段的数据。SAS 在读取完第 1~6 栏的数据后，就会读取后面的数据，因此不会影响到之后要读取的正确字段。具体程序如下，生成的对应表格如图 1-7 所示。

```
data a;
input y 1-4 m month 1-6 id a $ b c;
datalines;
199601 1101 a 2 3
199602 1102 b 3 4
199603 1103 d 2 4
;
```

	y	m	month	id	a	b	c
1	1996	1	199601	1101	a	2	3
2	1996	2	199602	1102	b	3	4
3	1996	3	199603	1103	d	2	4

图 1-7

2. 读取记事本数据

本质上讲，读取记事本数据的方法和直接将数据粘贴在程序中的操作方法是一样的，读者需要根据数据的形式和字段确定读取的方式，相关程序如下所示：

```
data price;
infile 'D:\The Application of SAS in Financial Research\Raw data\monthly
price\d2007' firstobs=3;
input y 1-8 m code name $ volum value ret turn mv PE PB high low;
/*年 月 日 股票代码 股票名称 成交量 成交值 报酬率 市值 市盈率 市价净值比 最高价 最低
价*/
run;
```

上述程序中，SAS 不会执行写在 “/**/” 之间的命令，该字段作用是提示，是用户自行对程序所下的脚注、说明，是协助用户使用程序时的好帮手，如表 1-4 所示。

表 1-4

语 法	含 义
/**/	SAS 的注记，一般初学程序时一定要学会一边写程序一边注明写这段程序的原因，可用于提醒
data	告知 SAS 最后生成的 table 要命名为 price，开头不得为数字
infile	告知 SAS 要读取的文本文件的路径及文件名
firstobs	为 infile 的一个特别语法，告知 SAS 从第几个观测值开始读取
input	告知 SAS 输入的变量名称，必须为英文变量名
\$	在变量后加入\$，告知 SAS 该变量为文本格式

夹在两个单引号间的语法表示的是数据来自于外部，用途是声明路径，告知 SAS 如何搜寻目标文件夹或文档，当然前提是该路径是存在的。指令的结果是读取 d2007 的外部文档，并在 SAS 里生成一个名为 price 的文档，如图 1-8 所示。请注意文件名只能由英文、数字组成，不得为中文。

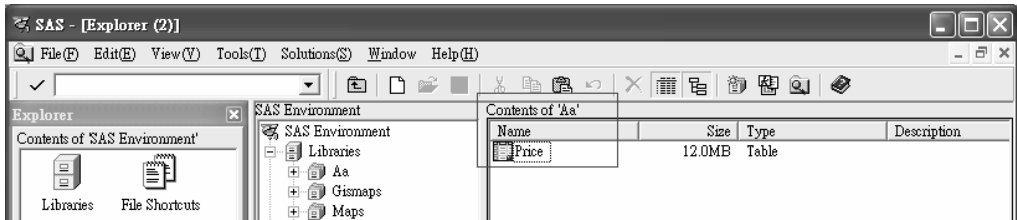


图 1-8

该例中的数据包含股票的每月成交量（百万股）、每月成交值（百万元）、每月报酬率、每月市值、每月市盈率、每月股价净值比、每月最高价和每月最低价。该数据经过除权息调整后，并非是投资者每天看到的真实最高、最低价，但仍可作为程序解说等用途。

虽然本部分只简单介绍了记事本数据的读取程序。目前财务数据库通常都提供了建好的 SAS 文件，如国泰安数据库，在下载数据时提供了读取该数据的程序文件。因此读者读取数据时，不需要花费太多功夫。

3. 输入 Excel 文件数据

SAS 可读取 Excel 文件，在 SAS 9.2 的版本中，SAS 能处理 Office 2003 以下的版本，所以 Office 2007 的使用者，必须先将目标 Excel 文件数据转存成 Excel 2003 才能使用。下面介绍如何使用 SAS 来读取外部 Excel 文件格式，要输入 Excel 文件，需要使用 SAS 内建的 proc import，通过该程序可以将 Excel 的工作表（sheet）转成 SAS 中的 table。

```
proc import datafile='D:\The Application of SAS in Financial Research\Raw
data\event date\foreign'
  dbms=xls
  out=event
  replace;
```

```
sheet="sheet1";
run;
proc import datafile='D:\The Application of SAS in Financial Research\Raw
data\event date\foreign for.xlsx'
dbms=xlsx
out=event_xlsx
replace;
sheet='sheet1';
run;
proc import datafile='D:\The Application of SAS in Financial Research\Raw
data\event date\foreign for.csv.csv'
/*CSV 文档格式不需要 dbms 的语法
且副文档名要标注在文档名之后
foreign for.csv.csv*/
out=event_csv
replace;
*sheet='sheet1';
*CSV 只有一个工作表，此时可以不写 sheet 的语法；
run;
```

上述程序中有以下几个重要语法，如表 1-5 所示。

表 1-5

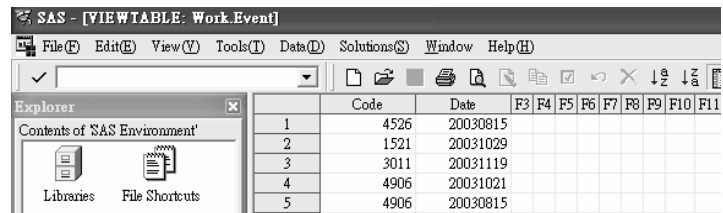
语 法	含 义
datafile	告知 SAS 经由哪条路径读取文件
out	告知 SAS 将要输出的文件名
dbms	告知 SAS 读取目标文件的副拓展名，若是 Excel 2003 以前版本的数据，输入 xls，若是 Excel 2007 的格式，则输入xlsx
replace	告知 SAS 取代之前已经有相同文件名的 table
sheet	告知 SAS 读取哪个工作表，若该 Excel 文件只有一个工作表，则可以忽略不写，若有多个工作表，则必须指定

执行上述程序，就会在 SAS 系统中生成 event、event_xlsx 和 event_csv 三种文件。需要注意的是 CSV 格式的文件打开方式虽然与 Excel 文档一样，但是其不像 Excel 有多个工作表（sheet），因此读取方式和 xls、xlsx 的方式稍有不同。另外在使用 proc import 的语法时，读取的目标文档应处于关闭状态，否则会造成程序运行失败。

读取 Excel 有时候会将非目标的单元格数据（cell）读进来，这会影响之后分析的准确性，所以在读取 Excel 文件前，要先打开 table 查看是否有不需要的变量。

如图 1-9 所示，event 文件中除了有 code 和 date 两个变量，还有完全空白的 F3、F4、……、F11 等变量¹，这是 SAS 将 Excel 中的其他字段读进来了，而 code 以及 date 两个变量字段，在 Excel 文件里面已经编好，与读取记事本时在读取中命名变量不同，读取 Excel 的外部文件，是要在 Excel 里面先对变量进行命名的。

1 若使用 SAS 9.4 版本读取，会出现 C、D、E、F 等 Excel 的栏位名称。

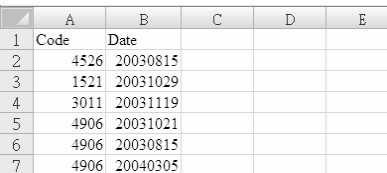


The screenshot shows the SAS interface with a window titled 'SAS - [VIEWTABLE: Work.Event]'. The main data table has the following structure:

	Code	Date	F3	F4	F5	F6	F7	F8	F9	F10	F11
1	4526	20030815									
2	1521	20031029									
3	3011	20031119									
4	4906	20031021									
5	4906	20030815									

图 1-9

如图 1-10 所示，code 和 date 在 Excel 工作表中的命名已经完成，而且未出现 F3 到 F11 等变量的名称，那么针对 F3 到 F11 等变量，就需要先将其删除。首先，从 Excel 的容量开始，以竖列为例，Excel 可以容纳的变量共有 255 个，所以 SAS 读取 Excel 的变量不会超过 255 个，而横列的观测值无法超过 65535 个，熟记该特性后，我们可以进行如下程序操作。



The screenshot shows an Excel spreadsheet with the following data:

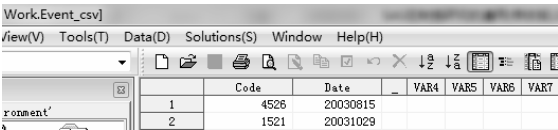
	A	B	C	D	E
1	Code	Date			
2	4526	20030815			
3	1521	20031029			
4	3011	20031119			
5	4906	20031021			
6	4906	20030815			
7	4906	20040305			

图 1-10

```
proc import datafile="D:\SAS 在财务研究上的运用\Raw data\事件日\外资.xls"
out=event2 (drop=f1-f255)
replace;
sheet="sheet1";
run;
```

上述程序增加了“(drop=f1-f255)”告知 SAS 系统将这些变量删除，而这会造成 log 文件上出现绿色警告 (warning)，原因是 f1-f255 这 255 个变量不会完全存在。因此在命名 Excel 变量时，不能将变量名称命名为 F1-F255，以免影响之后的数据处理。

读取 CSV 文件时，会出现如图 1-11 所示的结果。



The screenshot shows the SAS interface with a window titled 'Work.Event_csv'. The main data table has the following structure:

	Code	Date	VAR4	VAR5	VAR6	VAR7
1	4526	20030815				
2	1521	20031029				

图 1-11

读取 CSV 格式的文件时，可采用保留变量的语法来进行操作，相关程序如下所示：

```
proc import datafile='D:\The Application of SAS in Financial Research\Raw
data\event date\foreign for csv.csv'
out=event3 (keep=code date)
replace;
run;
```

有些读者担心如果读取的 Excel 的变量过多，在撰写“keep=”后面的变量组时可能会很繁杂。实际上这时只要打开 Excel，复制变量栏并粘贴到 SAS 程序中编辑即可，如图 1-12 所示，相关语法介绍如表 1-6 所示。

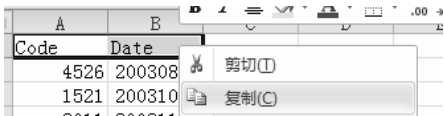


图 1-12

表 1-6

语 法	含 义
(drop=varlist ¹)	在声明 SAS 的表格（读取或输出皆可）时使用，会删除一组变量
(keep=varlist)	在声明 SAS 的表格（读取或输出皆可）时使用，会保留一组变量

（drop=varlist）和（keep=varlist）的使用时机是在读取数据或输出数据时对表格做的第一步（读取时）或最后一步（输出时）处理。

4. 读取外部 SAS 文件

获取分析数据时，除了使用记事本数据和 Excel 数据外，也有可能使用已经建好的 table 文件，例如国外常见的 CRSP、COMPUSTAT 以及 IBES 数据库直接提供的 SAS 格式的文件，此时该如何处理？

如图 1-12 所示，在”D:\The Application of SAS in Financial Research\SAS data\daily data”的路径下，有 2001 年至 2006 年某上市公司日报酬率数据，该如何读取这些报酬率数据呢？



图 1-12

具体操作代码如下所示：

```
data a;  
  set 'D:\The Application of SAS in Financial Research\SAS data\daily  
data\d2002';  
run;
```

1 varlist 指的是变量组（variable list），为行文方便，本书在 SAS 语法中以 varlist 替代中文的变量组。

使用上述方法可以直接读取文件夹里 d2002 的 SAS 文件。但如果要读完 d2001~d2006 文件呢？重复输入 “D:\The Application of SAS in Financial Research\SAS data\daily data” 的操作虽然仅需 “复制、粘贴”，但可能夹杂其他文件夹的数据，给结果分析带来很大不便。有另一种方法可以帮助解决这样的问题。

我们可以通过 SAS 打开目标文件夹，然后做进一步的处理，相关程序如下所示：

```
libname aa 'D:\The Application of SAS in Financial Research\SAS data\daily data';
```

相关语法介绍如表 1-7 所示。

表 1-7

语 法	含 义
libname	创建一个 SAS 逻辑库
aa	文件夹名称，为方便起见命名为 “aa”，也可以有其他命名，如 “stock”、“return”，虽然这样很直观，但在之后使用上较为不便，所以建议文件名应尽量简单
路径	文件夹的路径所在

在执行程序前，“aa” 这个 SAS 文件夹是不存在的，而是有图 1-13 所示的五个逻辑库。

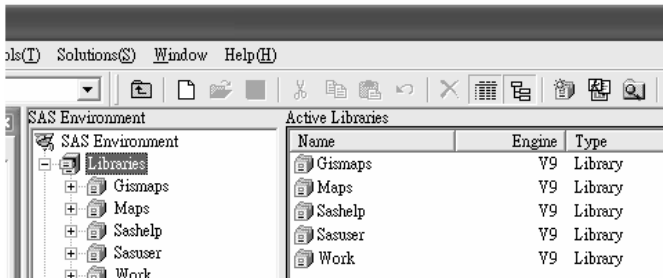


图 1-13

在这五个逻辑库中，所有执行的 table 都是在 “work” 里面进行的，“work” 也是其中最重要的逻辑库，执行程序之后，会有以下结果：

Name	Engi...	Type	Host Pathname
Aa	V9	Library	D:\The Application of SAS in Financial Research\SAS data\daily data

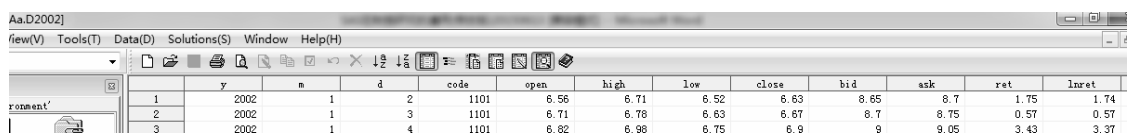
在使用外部 SAS 数据文件时，切记不要将原有的 SAS 文件覆盖掉，建议采用下面的方法读取，且文件夹命名要注意与已经存在的五个逻辑库区分开。这样 “table a” 的数据就是 “aa” 逻辑库里的 d2002。

```
data a;  
set aa.d2002;  
run;
```

相关语法介绍如表 1-8 所示。

表 1-8

语 法	含 义
set	读取其后的数据，亦可读取多个 table，但要注意若是读取多个文件，则第二个文件的数据会在第一个文件数据的下面，依次类推
aa.d2002	“aa.” 是告知 SAS 读取 “aa” 这个逻辑库，不写 aa. 则 SAS 会直接读取 work 逻辑库里的文件，亦即读取 work.d2002，但由于 work 是 SAS 内建的临时逻辑库，初始设定 set d2002 等同于 set work.d2002



	y	m	d	code	open	high	low	close	bid	ask	ret	lnret
1	2002	1	2	1101	6.56	6.71	6.52	6.63	8.65	8.7	1.75	1.74
2	2002	1	3	1101	6.71	6.78	6.63	6.67	8.7	8.75	0.57	0.57
3	2002	1	4	1101	6.82	6.98	6.75	6.9	9	9.05	3.43	3.37

在 d2002 的数据中包含了许多变量，在处理大数据时，若读者事先知道自己仅需要 y、m、d、code、ret 这几个变量，则可以利用（keep=y m d code ret）的语法，让 SAS 只读取这些变量，可加快运行速度。可通过以下两种方法进行操作。

方法一：

```
data a;  
set aa.d2002(keep= y m d code ret);  
run;  
/*在读取数据时，只读 y m d code ret 运行速度较快*/
```

方法二：

```
data a(keep= y m d code ret);  
set aa.d2002;  
run;  
/*将 aa.d2002 完整读取后 保留 y m d code ret 运行速度较慢*/
```

本节简单介绍了如何将数据读取到 SAS 中，基本上可以很顺利地读取记事本数据、Excel 数据及外部的 SAS 文件，方便日后的程序撰写和统计处理。

1.4 如何输出数据

部分用户将 SAS 视为数据处理中间环节的工具，在处理完相关数据以后，常常还需要用其他软件做后续计算；而一些学术研究者有时需要对统计结果进行交叉验证，这时用其他软件可能无法读取 SAS 的 table 文件。

本节会介绍如何将 SAS 的 table 文件转存成记事本数据、Excel 文件及外部逻辑库的 table 文件。

1. 输出为记事本数据

```
libname aa 'D:\The Application of SAS in Financial Research\SAS data\daily  
data\';  
data a;
```

```
set aa.d2002;
run;
proc export data=a
outfile="D:\The Application of SAS in Financial Research\CH01\output\a.txt"
replace;
run;
```

上述程序运用了 proc export 语法，该程序语法除了可以用于输出形成记事本数据外，亦可将 table 文件输出转成 Excel 格式，相关语法介绍如表 1-9 所示。

表 1-9

语 法	含 义
outfile	将文件输出到后面路径中的文件名
a.txt	将文件输出为 txt 格式，前面为该文件所属的路径，用双引号对路径文件进行标注
replace	若该路径下已有相同文件名的文件，将其取代

为了简单起见，将文件名命名为 a，使用者亦可根据自己的需要更改文件名。若 SAS 使用者的编辑器是繁体中文格式，则文件名称可以用中文命名，此处要特别注意一点，使用 SAS 将数据输出转成记事本格式后，之后读取数据时不能使用字段读取，重新读取需要直接读取变量，不能计算字段。

2. 输出成 Excel 数据

```
proc export data=a
outfile='D:\The Application of SAS in Financial Research\CH01\output\报酬率'
dbms=xlsx
replace;
sheet="y2002";
/*一定要指定输出的 sheet
否则 SAS 以报酬率作为 sheet 取代掉
*/
run;
```

从本质上讲，输出转成 Excel 格式的程序语法结构与输出转成记事本格式的语法相似，不同的是 Excel 格式多了工作表的分隔，还需要对工作表进行命名。相关语法介绍如表 1-10 所示。

表 1-10

语 法	含 义
outfile	将文件输出到后面路径中的文件名
报酬率.xls	将文件输出为 Excel 格式，前面为该文件所属的路径，用双引号进行标注
replace	若该路径下已有相同文件名的文件，将其取代
Sheet	告知 SAS 工作表名称，若不声明，SAS 会自动将工作表命名为输出的 table 名称（在本例中，会命名为 a）

要将table输出成Excel文件仍然要注意数据容量的限制,由于Excel最多只能容纳255个变量和65536个观测值，其中第一个观测值必须命名为变量，所以最多只能有 65535 个观测值，用户在撰写程序时，

切记不要输出超过 65535 笔数据¹。此外，我们将其输出成 CSV 文件则无此限制。相关程序如下所示：

```
/*
基本上 Excel2003 文件有 65535 笔数据的限制
    Excel 2007 文件大约有一百多万笔数据的限制
可以输出成 CSV 格式
其数据处理如同 Excel 文件
*/
proc export data=a
outfile="D:\The Application of SAS in Financial Research\CH01\output\报酬率.csv"
replace;
run;
```

虽然输出转成 CSV 文件可以不受 65535 个数据限制，且文件大小也比 Excel 的小，但 CSV 文件不像 Excel 文件一样可以打开不同的工作表（sheet），每个文件仅能存取一个工作表。因此可以根据自己的需求，将数据输出成不同形式。

3. 转存成外部的 SAS 文件

SAS 转存数据作为外部文件有两种方法：第一种是直接撰写语法处理，第二种是声明外部逻辑库后转存。

```
方法一：
data 'D:\The Application of SAS in Financial Research\CH01\output\b';
set a;
run;

方法二：
libname bb 'D:\The Application of SAS in Financial Research\CH01\output';
data bb.a;
set a;
run;
```

要将 SAS 存在计算机里的永久逻辑库，较佳的方法就是声明新的逻辑库，然后将文件名存出来，当然将记事本以及 Excel 的数据读入到 SAS 时，也可以做相同的处理，亦即在文件名前面增加 aa、bb、... 等相关声明后的逻辑库，使用者要切记如果没有加 aa 或者 bb 的逻辑库，等同于处理 work 文件。

1.5 总结

本章主要帮助初级 SAS 使用者了解 SAS 的一些基本接口，并简单介绍如何读取数据，如何将数据输出转成记事本数据和 Excel 数据，其中还穿插着与数据处理和输出相关的建议。

¹ 如果数据超过 65535 笔，可将数据拆成两个文件后再输出。

第 2 章

SAS 数据的运算与函数

准备好 SAS 的数据之后，便要开始进行数据的运算整理。例如，报酬率数据为 10.25%，在数据数据中可能以 10.25 保存，若研究中要使用报酬率的平方，应该是以 0.1025 来计算平方；即无法直接使用 10.25% 进行计算，故必须先对该数据进行处理。

2.1 四则运算

本节简单使用变量中的四则运算来演示 SAS 中的程序写法。首先将已经准备好的数据读取进来，如图 2-1 所示，该文件包含 3 个变量和 20 笔观测值。相关程序代码如下：

	a	b	c
1	28	17	17
2	0	11	-24
3	15	4	5
4	10	-7	2
5	16	-1	12
6	2	-4	-8
7	-3	-9	-7
8	24	10	7
9	0	-20	-19
10	13	3	2
11	3	-12	-4
12	9	8	-12
13	21	8	8
14	8	-14	30
15	5	2	-1
16	5	3	-12
17	22	-1	7
18	9	-12	-2
19	1	-9	-7
20	17	8	8

图 2-1

```
libname aa 'D:\The Application of SAS in Financial Research\CH02\data';
data a;
set aa.a;
run;
data b;
set a;
d=a+b;
e=c-b;
f=a*b;
g=b/d;
h=b**2;
i=4;
j=i**(0.5);
run;
```

上述程序进行了几种运算，分别是加、减、乘、除、平方和开根号，相关语法含义如表 2-1 所示。

表 2-1

语 法	含 义
d=a+b	变量 d 等于变量 a 加上变量 b
e=c-b	变量 e 等于变量 c 减去变量 b
f=a*b	变量 f 等于变量 a 乘以变量 b

续表

语 法	含 义
g=b/d	变量 g 等于变量 b 除以变量 d
h=b**2	变量 h 等于变量 b 的平方，在 SAS 中 2**3 表示 2 的 3 次方，若为 2** (1/3) 则表示为 2 开 3 次方
i=4	变量 i 等于 4
h=i**(0.5)	变量 h 等于变量 i 开根号

这里要特别注意的是变量 g 等于 b/d，d 在原本的 SAS 文档里并不存在，这也显示出 SAS 执行程序的顺序，虽然原始文件中并不存在变量 d，但在执行 g=b/d 这步之前，系统已经先运算好 d 这个变量的结果了，故不会影响到后续程序的执行。另外，i=4 是另一个有趣的指令，是告知 SAS 对所有的观测值处理时，变量 i 都等于 4。上述程序执行后的结果如图 2-2 所示。

a	b	c	d	e	f	g	h	i	j
28	17	17	45	0	476	0.3777777778	289	4	2
0	11	-24	11	-35	0	1	121	4	2
15	4	5	19	1	60	0.2105263158	16	4	2
10	-7	2	3	9	-70	-2.333333333	49	4	2
16	-1	12	15	13	-16	-0.066666667	1	4	2
2	-4	-8	-2	-4	-8	2	16	4	2
-3	-9	-7	-12	2	27	0.75	81	4	2
24	10	7	34	-3	240	0.2941176471	100	4	2
0	-20	-19	-20	1	0	1	400	4	2
13	3	2	16	-1	39	0.1875	9	4	2
3	-12	-4	-9	8	-36	1.333333333	144	4	2
9	8	-12	17	-20	72	0.4705882353	64	4	2
21	8	8	29	0	168	0.275862069	64	4	2
8	-14	30	-6	44	-112	2.333333333	196	4	2
5	2	-1	7	-3	10	0.2857142857	4	4	2
5	3	-12	8	-15	15	0.375	9	4	2
22	-1	7	21	8	-22	-0.047619048	1	4	2
9	-12	-2	-3	10	-108	4	144	4	2
1	-9	-7	-8	2	-9	1.125	81	4	2
17	8	8	25	0	136	0.32	64	4	2

图 2-2

观察上述执行结果，发现运算后若除不尽，SAS 会计算到小数点后 10 位，这会给阅读、分析数据带来不便，因此需要进行进一步的整理，以使数据看起来比较清晰。

2.2 统计函数

下面我们利用前一节的文档做进一步的处理，相关程序如下：

```
data c;
set b;
a1=int(g);
a2=mod(f,10);
a3=round(g,0.01);
a4=abs(g);
a5=round(abs(g),0.01);
a6=exp(f);
```

```

a7=log(10);
a8=log10(10);
a9=lag(a);
a10=lag2(a);
a11=dif(a);
a12=dif2(a);
run;

```

上述程序中有几个常见的数学函数，相关语法含义如表 2-2 所示。

表 2-2

语 法	含 义
a1=int(g)	变量 a1 等于变量 g 的整数部分
a2=mod(f,10)	变量 a2 等于变量 f 除以 10 的余数
a3=round(g,0.01)	变量 a3 等于变量 g 四舍五入到小数点后 2 位
a4=abs(g)	变量 a4 等于变量 g 的绝对值
a5=round(abs(g),0.01)	变量 a5 等于变量 g 取绝对值后四舍五入到小数点后 2 位
a6=exp(f)	变量 a6 等于 e^f ，亦即 e 的 f 次方
a7=log(10)	变量 a7 等于 10 的自然对数
a8=log10(10)	变量 a8 等于 10 取 10 的自然对数
a9=lag(a)	变量 a9 等于变量 a 滞后 1 期的数据
a10=lag2(a)	变量 a10 等于变量 a 滞后 2 期的数据
a11=dif(a)	变量 a11 等于变量 a 减去 lag(a)，亦即与前 1 期的差分
a12=dif2(a)	变量 a12 等于变量 a 减去 lag2(a)，亦即与前 2 期的差分

用户可以利用这些函数进行其他运算，假设变量 a 是股价，那么变量 a11 除以 a9 就是报酬率。在撰写 SAS 程序时，需要掌握使用最基本的函数进行其他复杂的组合运算，例如 a5 便是将四舍五入进位的函数与取绝对值函数组合在一起。接下来检验程序执行后的结果，如图 2-3 所示。

a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12
0	6	0.38	0.3777777778	0.38	5.298749E206	2.302585093	1				
1	0	1	1	1	1	2.302585093	1	28		-28	
0	0	0.21	0.2105263158	0.21	1.1420074E26	2.302585093	1	0	28	15	-13
-2	0	-2.33	2.3333333333	2.33	3.97545E-31	2.302585093	1	15	0	-5	10
0	-6	-0.07	0.0666666667	0.07	1.1253517E-7	2.302585093	1	10	15	6	1
2	-8	2	2	2	0.0003354626	2.302585093	1	16	10	-14	-8
0	7	0.75	0.75	0.75	532048240602	2.302585093	1	2	16	-5	-19
0	0	0.29	0.2941176471	0.29	1.700888E104	2.302585093	1	-3	2	27	22
1	0	1	1	1	1	2.302585093	1	24	-3	-24	3
0	9	0.19	0.1875	0.19	8.65934E16	2.302585093	1	0	24	13	-11
1	-6	1.33	1.3333333333	1.33	2.319523E-16	2.302585093	1	13	0	-10	3
0	2	0.47	0.4705882353	0.47	1.8586717E31	2.302585093	1	3	13	6	-4
0	8	0.28	0.275862069	0.28	9.1510928E72	2.302585093	1	9	3	12	18
2	-2	2.33	2.3333333333	2.33	2.285694E-49	2.302585093	1	21	9	-13	-1
0	0	0.29	0.2857142857	0.29	22026.465795	2.302585093	1	8	21	-3	-16
0	5	0.38	0.375	0.38	3269017.3725	2.302585093	1	5	8	0	-3
0	-2	-0.05	0.0476190476	0.05	2.789468E-10	2.302585093	1	5	5	17	17
4	-8	4	4	4	1.247946E-47	2.302585093	1	22	5	-13	4
1	-9	1.13	1.125	1.13	0.0001234098	2.302585093	1	9	22	-8	-21
0	6	0.32	0.32	0.32	1.1589095E59	2.302585093	1	1	9	16	8

图 2-3

在撰写 lag 与 dif 函数时，需要特别注意的是，如果使用的数据是多公司数据或者多国数据，会出现前一家公司的最后一笔数据，变成后一家公司第一笔数据的滞后一期的结果，此时则需要进行调整，有关内容会在后面进行介绍。

在处理数据的过程中，有时候需要做一些变量的描述统计，SAS 提供了针对每个观测值进行描述统计的函数，相关程序如下所示：

```
data a;
set aa.b;
a1=mean(col1,col2,col3,col4,col5)-1;
a2=mean(of col1-col4)-1 ;
a3=median(of col1-col4);
a4=var(of col1-col4);
a5=std(of col1-col4);
a6=stderr(of col1-col4);
a7=geomean(of col1-col3)-1;
a8=(col1*col2*col3)**(1/3)-1;
drop col1-col12 ;
run;
```

这里采用的是虚拟出来的公司股票报酬率的数据，利用转置的方式将数据进行排列。每个观测值都有从 col1 到 col12 的数据，分别为 1 月到 12 月的报酬率，为了后续计算的便利，我们使用报酬率的原始值加上 1，若股票报酬率为 10%，则该值为 1.1，上述程序出现了均值（mean）、中位数（median）、方差（var）、标准偏差（std）、标准误（stderr）和几何均值（geomean）的语法。在这些函数中，笔者认为几何均值是最重要的语法，因为财务上计算平均报酬率较佳的方式是使用几何平均法，例如一只股票先涨 100%，再跌 50%，就简单的算术平均值来看是 25%，但投资者的实际报酬率为 0。运行上述程序生成的结果如图 2-4 所示。

code	y	NAME OF FORMER VARIABLE	a1	a2	a3	a4	a5	a6	a7	a8
1101	2001	ret	0.03	0.02	1.02	0.0004	0.02	0.0115470054	0.0299514552	.
1101	2002	ret	0.0333333333	0.03	1.03	0.0002	0.0141421356	0.01	0.02	.
1101	2003	ret	-0.024	-0.015	0.965	0.0019	0.0435889894	0.0217944947	-0.00746898	-0.00746898
1102	2001	ret	-0.004	-0.02	0.96	0.0022	0.0469041576	0.0234520788	-0.043344975	-0.043344975
1102	2002	ret	-0.006	0.005	0.995	0.0009666667	0.0310912635	0.0155456318	0.0095746992	0.0095746992
1102	2003	ret	0.002	0.0025	1.01	0.0011583333	0.0340342964	0.0170171482	0.0164902438	0.0164902438

图 2-4

图中 a7 与 a8 的数值有些许的差异，此原因来自于采用非 SAS 内建的函数计算时，只要计算的数值中含有一个缺值，计算结果就会显现为缺值。在原始数据的设计中，第 1 笔与第 2 笔被设计含有缺值，因此产生了这样的结果。本节介绍的统计函数都有一个特性，就是针对非缺值的变量计算统计量，缺值部分直接忽略不计。

下面介绍有效观测值（n）、缺值数目（nmiss）、最小值（min）、最大值（max）、偏态系数（skewness）、峰态系数（kurtosis）及总和（sum）的用法。相关程序如下所示，运行结果如图 2-5 所示。

```
data a;
set aa.b;
```

```

a1=n(of coll-coll2);
a2=nmiss(of coll-coll2);
a3=min(of coll-coll4);
a4=max(of coll-coll4);
a5=skewness(of coll-coll4);
a6=kurtosis(of coll-coll2);
a7=sum(of coll-coll4);
a8=n(of coll-coll4);
a9=a7/a8;
drop coll-coll2;
run;

```

code	y	NAME OF FORMER VARIABLE	a1	a2	a3	a4	a5	a6	a7	a8	a9
1101	2001	ret	11	1	1	1.04	0	-1.25486087	3.06	3	1.02
1101	2002	ret	10	2	1.02	1.04	.	-0.40962206	2.06	2	1.03
1101	2003	ret	12	0	0.96	1.05	1.931921969	-1.50345029	3.94	4	0.985
1102	2001	ret	12	0	0.95	1.05	1.938188331	-1.74055017	3.92	4	0.98
1102	2002	ret	12	0	0.98	1.05	1.597077983	-0.95492183	4.02	4	1.005
1102	2003	ret	12	0	0.96	1.03	-0.62780515	-1.54387876	4.01	4	1.0025

图 2-5

本节介绍的描述统计函数，除了 geomean 外，都可以使用 SAS 的程序步实现。最后注意一点，在计算投资组合的报酬率时，应该将数据形式转换成本范例的数据格式，以迅速求得持有期间的平均报酬率。

2.3 随机函数

随机函数的基本作用是产生数据。研究者在进行理论研究或者推导的时候，未必会拥有实际数据，因此需要利用仿真出来的数据验证自己的理论。相关函数介绍如表 2-3 所示。

表 2-3

函 数	说 明
ranbin(seed,n,p)	函数 RANBIN 根据给定的 seed,n,p 传回数值，该数值符合为均值为 np、方差为 np(1-p)的二项式分布的随机数值
rancau(seed,x)	函数 RANCAU 根据给定的 seed，会传回一位置参数为 0， $\gamma=1$ 的标准科西分布的随机数值
ranexp(seed)	RANEXP 根据给定的 seed，传回方差为 1 的自然指数分布的随机数值
rangam(seed,a)	RANGAM 根据给定的 seed, a，传回形状参数为 a 的伽玛分布的随机数值
rannor (seed)	RANNOR 据给定的 seed，传回均值为 0,方差为 1 的标准正态分布的随机数值
ranpoi (seed,m)	RANPOI 据给定的 seed,m,传为均值为 m 的卜瓦松分布的随机数值
ranuni (seed)	RANUNI 据给定的 seed 传为数值介于 (0,1) 之间的均等分布的随机数值

SAS 内建的随机函数包含二项 (binomial) 分布、柯西 (Cauchy) 分布、自然指数 (exponential) 分布、伽马 (Gamma) 分布、标准正态 (normal) 分布、泊松 (poisson) 分布、均匀 (uniform) 分布等，这里仅摘录部分随机函数，进阶用户可自行查询 SAS 中的 “help”。

使用随机函数时，需要设置 seed，即给定 SAS 起始值。该起始值有几个要求，第一，该数值应介于 1 至 $2^{32}-1$ 之间；第二，必为整数；第三，该数值若为 0 或负值，SAS 会自动将起始值设置为程序执行开始的时间。

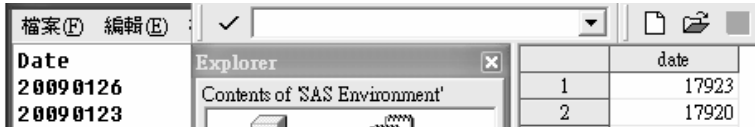
正确设置起始值的好处是不管何时执行程序，都可以得到相同的结果；缺点是仿真出来的数据结果会被质疑为只是通过试误法（trial and error）得到的较佳结果。而不设置起始值则无法确定程序最后执行的结果。

2.4 时间函数

SAS 中处理时间数据的相关程序，如下所示：

```
data date;
infile 'D:\SAS 在财务研究上的运用\CH02\data\date.txt' firstobs=2;
input date yymmddn8. ;
run;
```

在财务数据中，时间或日期方面的数据扮演着相当重要的角色，例如每年一月，股票绩效表现会出现比较好的“元月效应”。SAS 提供了一系列相关的时间函数来处理时间方面的问题，读取时间数据时，便要指定其为时间格式。行文至此，有读者或许会有疑问：为何第 1 章中处理月数据，在读取年份与月份时是分别读取两个变量，而此处读取日期时却只输入“yymmdd8.”？首先了解一下文本文件和 SAS 的 table 文档间的差异。



20090126	1	17923
20090123	2	17920

图 2-6

图 2-6 左边为记事本数据，右边为 SAS 读取数据后的结果。撰写程序时，刚接触 SAS 的使用者完全无法辨认右边文档的日期，这是因为 SAS 已经针对特定日期重新将每一天由 1 开始编码，想要确认日期必须通过时间函数重新转换。表 2-4 介绍了时间格式数据的读取方法。

表 2-4

语 法	含 义
yymmddn8.	数据为 19900125 的 8 位日期格式使用
yymmddn6.	数据为 900125 的 6 位日期格式使用
mmdddyn8.	数据为 01251990 的 8 位日期格式使用
mmdddyn6.	数据为 012590 的 6 位日期格式使用
ddmmyyn8.	数据为 25011990 的 8 位日期格式使用
ddmmyyn6.	数据为 250190 的 6 位日期格式使用

除了上述方法外，时间数据的表示形式还有 1990/01/25、1990-01-25 和 1990.01.25 等方法，在读取上也有所不同，当遇到这样的数据格式时，可以用表 2-5 所示的方法来读取。

表 2-5

语 法	含 义
yymmdd10.	数据为 1990/01/25 的 10 位日期格式使用
yymmddd10.	数据为 1990-01-25 的 10 位日期格式使用
yymmddp10.	数据为 1990.01.25 的 10 位日期格式使用

这里虽是 8 位数的日期介绍，但也适用于 6 位数格式的日期。如果是读取月份数据，则需要另外查询 SAS 的时间格式读取，本书建议使用者将年、月、日分开读取。下面是使用相关时间函数获取重要年、月、日、周、季等数据的程序：

```
data a;
set date;
y=year(date);
m=month(date);
d=day(date);
q=qtr(date);
w=week(date);
wday=weekday(date);
yyq=yyq(y,q);
newdate=mdy(m,d,y);
month=mdy(m,1,y);
run;
```

上述时间函数运用仅止于日数据，事实上 SAS 内建时间函数的最小单位为秒数，研究者如果需要使用日内数据，可查询 SAS 的 help，并在索引部分输入 MDY 查询，就能够查询到相关的时间函数用法，下面介绍这些时间函数的含义，如表 2-6 所示。

表 2-6

语 法	说 明
year(date)	求出时间格式的年份
month(date)	求出时间格式的月份
day(date)	求出时间格式的日
qtr(date)	求出时间格式的季度
week(date)	求出时间格式的周
weekday(date)	求出时间格式星期几，星期日为 1，星期一为 2，依次类推
yyq(y,q)	利用年份跟季度得到的时间格式数据
mdy(m,d,y)	利用月、日、年求出的时间格式数据
mdy(m,1,y)	利用月与年数据求出每月月初的时间格式数据

研究一下时间函数所取得的结果，采用时间格式的日期以及将其视为数字的变量的差别并不大，使用者可以任意将时间格式的日期转出年变量、月变量以及日变量，而时间格式的日期唯一的好处是可以将星期一到星期日进行编码，对星期效应有兴趣的人，有极佳的用途。图 2-7 为上述程序运行的结果¹。

	date	y	m	d	q	w	wday	yyq	newdate	month
1	17923	2009	1	26	1	4	2	17898	17923	17898
2	17920	2009	1	23	1	3	6	17898	17920	17898
3	17919	2009	1	22	1	3	5	17898	17919	17898
4	17918	2009	1	21	1	3	4	17898	17918	17898
5	17917	2009	1	20	1	3	3	17898	17917	17898
6	17913	2009	1	16	1	2	6	17898	17913	17898
7	17912	2009	1	15	1	2	5	17898	17912	17898
8	17911	2009	1	14	1	2	4	17898	17911	17898
9	17910	2009	1	13	1	2	3	17898	17910	17898

图 2-7

观察图 2-7 发现 newdate 变量的数值与 date 一样，这是“mdy 函数”将年月日转换后的结果，因此使用者不需要直接输入时间格式的变量，也可以轻松地利用时间函数进行转换，这可以省去研究正确的时间格式输入方法的时间。时间格式的变量在某些情况下是一定得使用的，例如绘制趋势图使用非时间格式的变量时，19900131 和 19900201 之间相隔将不是 1 天，而是 69 天，在提取跨年度数据时，该问题会突显得更加严重。

虽然时间函数存在不足之处，但 SAS 仍提供了一些格式化的语法，可以让时间格式的数据变成日期数据，相关程序如下所示：

```
data a;
set a;
format month yymon7. /*yymonw. w 的长度是 5 7 亦可写成 monyyw.*/
yyq yyq. /* yyqw. w 的长度可以是 4 5 6 10*/;
run;
data a;
set a;
format date yymmddd8. ;
/* yymmddxw mmdyyxw ddmmyyxw 年月日可以有三种显示形式
x 代表的是日期的输出格式 有 B C D N P S
w 为变量最后长度根据不同的输出格式有 2~10 种
但是并非所有的输出格式都能输出 2~10 位
*/
```

使用上述语法，可以将一些看似杂乱无章的数字转换成可读的时间数据，但建议还是使用一般的数字格式进行读取。而在绘图方面，可以上述语法画出时间序列的走势图。

时间函数中还有一个相当重要的语法需要单独介绍，使用日期格式变量的最大好处在于可以直接

1 事实上，时间格式的数据也可以进行格式的转换，以使数据看起来比较漂亮，具体操作详见 format 指令。

计算数据存在的时间长度。在财务数据中，常需要计算各家公司的存在时间，采用日期格式的变量可以快速计算一家公司的存续时间。

```
/*先产生一个特定的生日*/
data a;
do m=1 to 12;
do d=1 to 5;
birthday=mdy(2,1,1980);
date=mdy(m,d,1990);
if date^=. then output;
end;
end;
run;
```

在此处先虚拟一笔数据，分别为 1990 年 1 到 12 月的前 5 天，并且假设有一个人的生日为 1980 年的 2 月 1 日，SAS 用户如何在每一天的数据中求出此人已经出生了几年、几季、几月、几周甚至几天？SAS 中的 intck 可以帮助 SAS 用户解决这个问题，相关程序如下所示。

```
data a;
set a;
age=intck('year',birthday,date);
qtr=intck('qtr',birthday,date);
month=intck('month',birthday,date);
week=intck('week',birthday,date);
day=date-birthday;
run;
```

intck (‘frequency’, startday, endday)：在语法中依序输入目标频率、开始日期及结束日期即可。其中频率可以输入 “year”、“qtr”、“month”、“week” 和 “day”，但是输入 “day” 相当于直接用 endday 减去 startday。运行上述程序得出的结果，如图 2-8 所示。

m	d	birthday	date	age	qtr	month	week	day
1	1	7336	10958	10	40	119	518	3622
1	2	7336	10959	10	40	119	518	3623
1	3	7336	10960	10	40	119	518	3624
1	4	7336	10961	10	40	119	518	3625
1	5	7336	10962	10	40	119	518	3626
2	1	7336	10989	10	40	120	522	3653
2	2	7336	10990	10	40	120	522	3654
2	3	7336	10991	10	40	120	522	3655
2	4	7336	10992	10	40	120	523	3656
2	5	7336	10993	10	40	120	523	3657
3	1	7336	11017	10	40	121	526	3681
3	2	7336	11018	10	40	121	526	3682
3	3	7336	11019	10	40	121	526	3683
3	4	7336	11020	10	40	121	527	3684
3	5	7336	11021	10	40	121	527	3685
4	1	7336	11048	10	41	122	531	3712

图 2-8

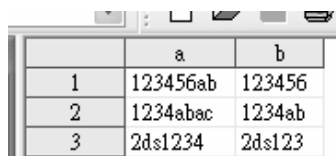
由结果可以发现，SAS 在 intck 函数的运算中，使用的是年分差直接相减，所以建议在计算年龄时可以使用出生总月份除以 12 来推算，否则可能会算出错误的公司年龄。

2.5 文本变量的处理

数据分析的过程中，有时候需要检验文本变量，或者对文本变量进行运算，读者可能会对此感到困惑：文本变量岂能像数字变量一样进行运算？事实上，文本变量可以进行一些运算或者处理，下面通过一些例子来介绍相关内容。

常见的数据库有 CRSP、Compustat、Datastream 及 IBES 等，进行数据分析时，需要将来源不同的数据合并在一起，而在其中两个数据库中，其相同的变量为 CUSIP，虽然如此，但是在 A 数据库中，该变量为 8 码，在另外一个数据库中，该变量为 6 位，其 6 位为 A 数据库 8 位数据的前 6 位，行文至此，读者可能会提出，将 8 位数据除以 100 取出其整数部位就可以了，但是该方法有其难度，以下利用一个假设的数据来介绍。

```
data a;
input a $ 1-8 b $ 1-6;
datalines;
123456ab
1234abac
2ds12348
;
run;
```



	a	b
1	123456ab	123456
2	1234abac	1234ab
3	2ds1234	2ds123

图 2-9

如图 2-9 所示，其代码数据是由文本与数字结合的文本变量，不可能运用数学函数进行运算。

SAS 中的 substr 函数可以将文本变量的数据直接提取出来，相关程序如下所示，运行结果如图 2-10 所示。

```
data a;
set a;
a1=substr(a,1,6);
a2=substr(a,6);
a3=substr(a,7,2);
run;
```

	a	b	a1	a2	a3
1	123456ab	123456	123456	6ab	ab
2	1234abac	1234ab	1234ab	bac	ac
3	2ds12348	2ds123	2ds123	348	48

图 2-10

由图 2-10 所示的结果来看，substr (a, 1, 6) 表示的是取出 a 变量的数据，由第 1 位开始读，读取 6 位数，substr (a, 6) 表示的是读取 a 变量第 6 位以后的所有字段的数据，substr (a, 7, 2) 表示由第 7 位开始读取，读取 2 位数，用 substr 获取数据似乎是可行的，但并非所有的数据都会像案例中的数据那样由左开始整齐排列¹。

```
data a;
input a $ 1-12 b $ 1-6;
datalines;
12345612
12341111
2ds12348
;
run;
data a;
set a;
a=right(a);
run;
```

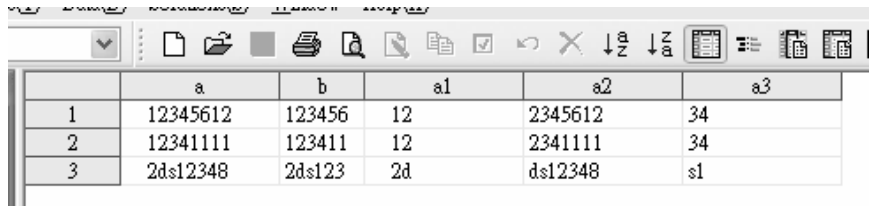
在本例的数据中，在数据后面多按了 4 次空格键，就数字变量而言，该空格键是不会有影响的，但是对于文本变量而言，“a”、“a ”以及“a ”这三种数据表面上都是 a，实际含义并不相同。为了使空白部分移到左边，此处加了一个文本变量的 right 函数，亦即要求 SAS 将数据由右开始排列，如图 2-11 所示。

	a	b
1	12345612	123456
2	12341111	123411
3	2ds12348	2ds123

图 2-11

就 SAS 的 table 文件来看，变量 a 的数据似乎并没有右对齐，但如果将光标移至变量 a 的单元格单击，就会发现数据“12345612”的前面有空格。而将光标移至变量 b 时，则没有显示前面的空格数据，如图 2-12 所示。接下来重新使用语法去萃取数据的变量，来思考遭遇到数据格式不完美的情况。

¹ 虽然在现行数据存储形式中，文字变量的字段都是由左开始向右排列的，但是由于操作疏失或其他因素，在获取数据的过程中，数据位数设定可能会有所差异。

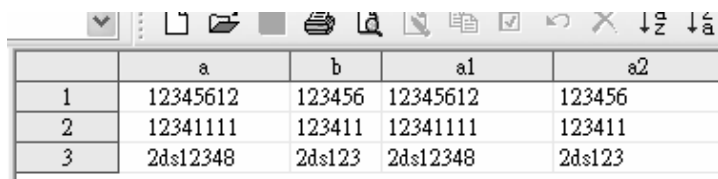


	a	b	a1	a2	a3
1	12345612	123456	12	2345612	34
2	12341111	123411	12	2341111	34
3	2ds12348	2ds123	2d	ds12348	s1

图 2-12

由数据类型来看，由于变量 a 数据前方有 4 个空白数据，运行 “substr (a, 1, 6)” 将无法取出正确的数据类型。此时可以采用下面的程序方法可以解决，运行结果如图 2-13 所示。

```
data a;  
set a;  
a1=left(a);  
a2=substr(a1,1,6);  
run;
```

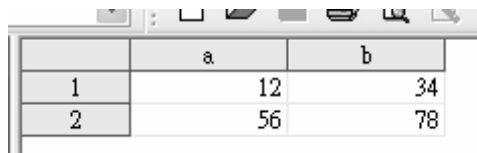


	a	b	a1	a2
1	12345612	123456	12345612	123456
2	12341111	123411	12341111	123411
3	2ds12348	2ds123	2ds12348	2ds123

图 2-13

如果读者在读取数据字段的过程中遇到问题，可以尝试使用 right 和 left 这两种函数来解决。接下来介绍文本变量的加法运算，相关程序如下所示，运行结果如图 2-14 所示。

```
data a;  
input a b;  
datalines;  
12 34  
56 78  
;  
run;
```



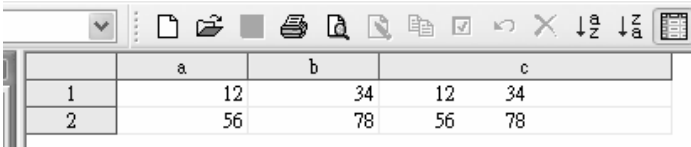
	a	b
1	12	34
2	56	78

图 2-14

首先产生两个数字变量，利用数字变量向右对齐的特性，可以进一步探讨一些文本变量的处理技巧。

```
data a;  
set a;  
c=a||b;  
run;
```

在此处，使用合并语法，将变量 a 与变量 b 合并，另外形成变量 c，该语法可以将不同变量任意合并在一起，不管合并的变量是文本变量还是数字变量，合并后的变量一律变为文本变量，运行结果如图 2-15 所示。



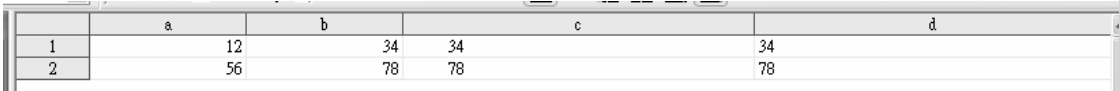
	a	b	c
1	12	34	12 34
2	56	78	56 78

图 2-15

结果发现，变量 c 的数据并非是 1234 连接在一起的，如果要将其连接在一起，可以使用下列的方法来处理。

```
data a;  
set a;  
c=b||'';  
d=left(c);  
run;
```

先将变量 b 与一个空白数据合并，将该变量转换成文本变量，再对该变量使用 left 函数，结果如图 2-16 所示。

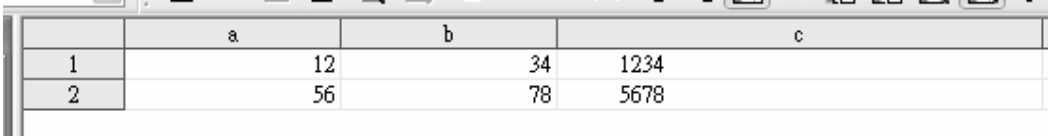


	a	b	c	d
1	12	34	34	34
2	56	78	78	78

图 2-16

```
data a;  
set a;  
c=a||d;  
run;
```

紧接着将变量 a 与变量 d 进行左右合并，如此就可以大致得到想要的结果，使 ab 两变量的数值紧密连接在一起，运行结果如图 2-17 所示。



	a	b	c
1	12	34	1234
2	56	78	5678

图 2-17

本例为了介绍程序语法，所以分开逐步撰写相关程序语法，其实可以使用复合性语法呈现¹。

```
data a;  
set a;  
c=left(a||left(b||''));  
run;
```

采用上面的复合语法，就可以直接得到需要的结果。相关语法介绍如表 2-7 所示。

表 2-7

语 法	含 义
substr	从文本变量中萃取出需要的位数
right	将文本变量向右对齐
left	将文本变量向左对齐
	将两个变量合并成一个新的文本变量

2.6 总结

本节主要介绍了 SAS 中执行运算的方法，如四则运算、统计函数、随机函数和时间函数，并简单阐述了如何对文本变量进行处理。用户在处理庞大的数据时，适时地运用相关函数功能可以提高效率。

1 本书中使用的语法常常会使用复合性的语法，或者呈现最佳的语法，但实际上都是通过逐步撰写后才重新更改的。

第 3 章

数据与变量的产生和选取

3.1 节为读者介绍如何利用 SAS 产生数据，3.2 节介绍保留变量和删除变量的指令，3.3 节介绍如何利用条件语句来保留或删除观测值，以及如何利用条件语句来指定新变量，3.4 节介绍抽样程序的相关内容。

3.1 利用 SAS 产生数据

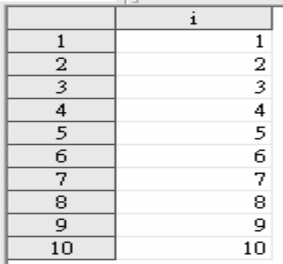
```
data a;  
do i=1 to 10;  
output;  
end;  
run;
```

该程序为产生数据的一个简单范例，在生成数据时通常会使用 do/end 的指令，该指令为 SAS 里的循环式指令，要求 SAS 由一个起始值开始进行，一直做到结束值。这样，操作者可以控制要生成的观测值数量。相关语法介绍如表 3-1 所示。

表 3-1

语 法	含 义
do i=1 to 10	从 1 做到 10
output	输出数据
end	结束
do 开始 to 结束 执行语法 end;	一个标准的循环式函数写法，do/end 之间夹杂的是要求要进行的指令函数

上例中的程序运行结果后得到的结果如图 3-1 所示。



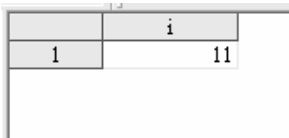
i	i
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10

图 3-1

上例中变量只有 i，其取值范围是从 1 到 10，如果 output 与 end 的顺序颠倒，那么运行结果是否会有所不同？

```
data b;  
do i=1 to 10;  
end;  
output;  
run;
```

将 output 写在 end 之后，即在 do 与 end 之间没有执行任何指令，结果将只会输出一笔观测值，如图 3-2 所示。所以切记 output 与 end 的先后顺序。

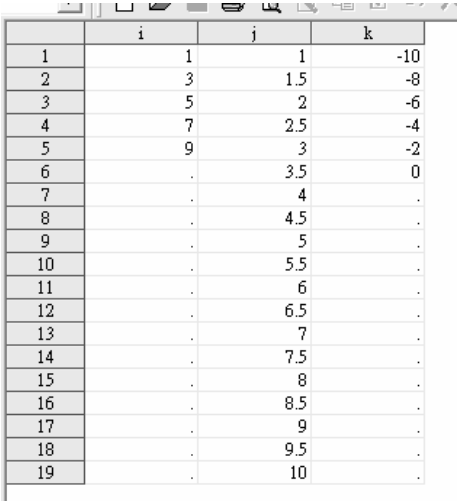


	i
1	11

图 3-2

```
data c;  
do i=1 to 10 by 2;  
output;  
end;  
run;  
data d;  
do i=1 to 10 by 0.5;  
output;  
end;  
run;  
data e;  
do i=-10 to 1 by 2;  
output;  
end;  
run;
```

上述程序分别加入了“by”的指令，表示在形成数据的过程中，分别以增加 2 和增加 0.5 的速度产生数据，执行后的结果如图 3-3 所示。



	i	j	k
1	1	1	-10
2	3	1.5	-8
3	5	2	-6
4	7	2.5	-4
5	9	3	-2
6	.	3.5	0
7	.	4	.
8	.	4.5	.
9	.	5	.
10	.	5.5	.
11	.	6	.
12	.	6.5	.
13	.	7	.
14	.	7.5	.
15	.	8	.
16	.	8.5	.
17	.	9	.
18	.	9.5	.
19	.	10	.

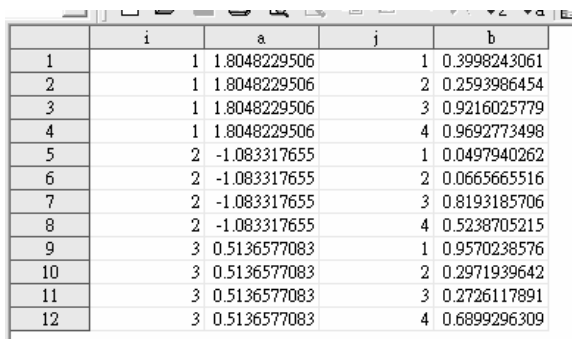
图 3-3

介绍了基本的数据产生语法后，接下来开始介绍结合随机数函数来产生仿真数据的语法。

```
data f;
do i=1 to 3;
a=rannor(1);
do j=1 to 4;
b=ranuni(2);
output;
end;
end;
run;
```

本例中的程序看起来稍微复杂了一点，指令中有 do i=1 to 3 和 do j=1 to 4，每个相对应的 do，都需要有一个相呼应的 end，否则 SAS 无法执行。

rannor 是执行产生标准正态分配的函数。本例中，a=rannor(1)指令位于 do i=1 to 3 和 do j=1 to 4 间，由于 SAS 执行指令是由上而下读取，而 a=rannor(1)又在 do j=1 to 4 之前，这表示 a=rannor(1)在 do i=1 to 3 之下只会执行 3 次，相同的 i 会有相同的 a 值。do j=1 to 4 虽然是独立的指令，但其被包含于 do i=1 to 3 指令下，表示每个 i 都会执行一次 do j=1 to 4 指令，因此 b=ranuni(2)这个均等分配函数会被重复执行 12 次，如图 3-4 所示。



	i	a	j	b
1	1	1.8048229506	1	0.3998243061
2	1	1.8048229506	2	0.2593986454
3	1	1.8048229506	3	0.9216025779
4	1	1.8048229506	4	0.9692773498
5	2	-1.083317655	1	0.0497940262
6	2	-1.083317655	2	0.0665665516
7	2	-1.083317655	3	0.8193185706
8	2	-1.083317655	4	0.5238705215
9	3	0.5136577083	1	0.9570238576
10	3	0.5136577083	2	0.2971939642
11	3	0.5136577083	3	0.2726117891
12	3	0.5136577083	4	0.6899296309

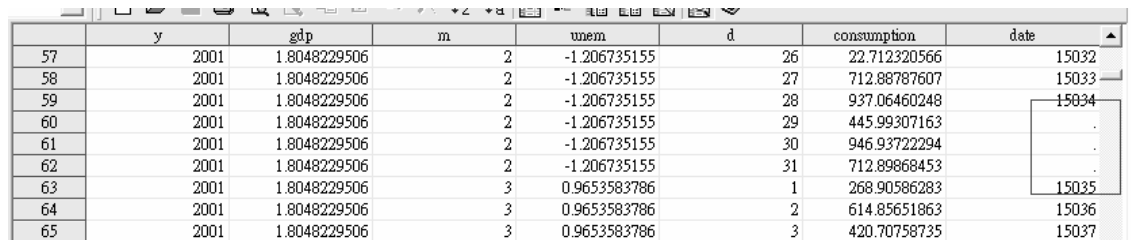
图 3-4

看到上面的结果，读者或许会困惑，我们不是要用随机数来产生数据吗？这样一来 a 跟 i 不就绝对相关了？不符合随机数完全随机的意义呀！因此接下来以较适合的参考变量帮助大家了解 a 变量在实际中不完全是随机的意义。

```
data g;
do y=2001 to 2005;
gdp=rannor(1);
do m= 1 to 12;
unem=rannor(2);
do d=1 to 31;
consumption=1000*ranuni(2);
date=mdy(m,d,y);
```

```
output;  
end;  
end;  
end;  
run;
```

本例中，GDP 为年度国内生产总值，unem 为月份失业率，ret 为报酬率，在这样的情况下，同一年度下每日的年度 GDP 数值必须要相同才行，unem 在同一个月份下也要一样。值得注意的是 do d=1 to 31，看似合理却又不正确，因为 2 月份不可能有 30 日与 31 日，此时 mdy 函数是否能起作用呢？程序执行后的结果如图 3-5 所示。由图可知 SAS 内建的时间函数对于实际不存在的日期不予编码，本章 3.3 节会介绍如何删除这样的数据，帮助研究者更好地建构 SAS 的日期数据。

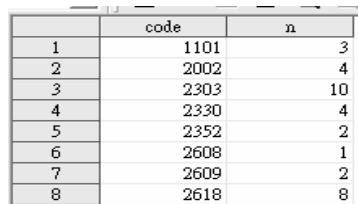


	y	gdp	m	unem	d	consumption	date
57	2001	1.8048229506	2	-1.206735155	26	22.712320566	15032
58	2001	1.8048229506	2	-1.206735155	27	712.88787607	15033
59	2001	1.8048229506	2	-1.206735155	28	937.06460248	15034
60	2001	1.8048229506	2	-1.206735155	29	445.99307163	.
61	2001	1.8048229506	2	-1.206735155	30	946.93722294	.
62	2001	1.8048229506	2	-1.206735155	31	712.89868453	.
63	2001	1.8048229506	3	0.9653583786	1	268.90586283	15035
64	2001	1.8048229506	3	0.9653583786	2	614.85651863	15036
65	2001	1.8048229506	3	0.9653583786	3	420.70758735	15037

图 3-5

```
data code1;  
  infile 'D:\The Application of SAS in Financial Research\CH03\data\code.txt'  
firstobs=2;  
  input code n;  
run;  
data code2;  
  set code1;  
  do n=1 to n;  
  a=rannor(1);  
  output;  
  end;  
run;
```

本例读取外部文件获取数据，并读入 code 和 n 两个变量，接着利用该文件里的 n 做循环指令，先看看 code1 文件的内容，共包含 8 笔观测值和 2 个变量，如图 3-6 所示。



	code	n
1	1101	3
2	2002	4
3	2303	10
4	2330	4
5	2352	2
6	2608	1
7	2609	2
8	2618	8

图 3-6

在 code2 循环指令 do n=1 to n 中, 针对每一个观测值执行了不同的 don=1 ton 的数据, 如图 3-7 所示。

	code	n	a
1	1101	1	1.8048229506
2	1101	2	-0.079915021
3	1101	3	0.396576855
4	2002	1	-1.083317655
5	2002	2	2.2382943651
6	2002	3	-0.624232294
7	2002	4	0.5136577083
8	2303	1	-0.086609117
9	2303	2	-0.594178733
10	2303	3	0.0318908181
11	2303	4	-0.737798572
12	2303	5	-0.250139175
13	2303	6	0.6850047647
14	2303	7	-0.804158132
15	2303	8	-0.74428108
16	2303	9	-0.795502822
17	2303	10	0.3407105493
18	2330	1	-0.300509804
19	2330	2	-1.349846516
20	2330	3	0.432704861
21	2330	4	1.3057162426
22	2352	1	1.4251270104
23	2352	2	-0.415801019
24	2608	1	1.6143805423
25	2609	1	-1.057726424
26	2609	2	-0.948332672
27	2618	1	0.95364764
28	2618	2	0.3919789416
29	2618	3	-0.076141279
30	2618	4	1.2205569287
31	2618	5	-0.630841419
32	2618	6	-0.635758247
33	2618	7	-0.340002833
34	2618	8	-0.076283637

图 3-7

3.2 保留、删除变量

```
data a;
do i=1 to 3;
a=ranuni(1);
do j=1 to 10;
b=rannor(1);
do k=1 to 10;
c=5+3*a+b*(-3)+rannor(2);
output;
end;
end;
end;
run;
```

利用上述程序产生一组 300 笔观测值的数据, 然后检查数据的情况以决定哪些变量要保留, 哪些变量要删除, 如图 3-8 所示。

	i	a	j	b	k	c
1	1	0.1849625698	1	-0.199218763	1	-119.4739616
2	1	0.1849625698	1	-0.199218763	2	-121.1622148
3	1	0.1849625698	1	-0.199218763	3	-119.8520116
4	1	0.1849625698	1	-0.199218763	4	-119.9393076
5	1	0.1849625698	1	-0.199218763	5	-120.2336073
6	1	0.1849625698	1	-0.199218763	6	-118.681737
7	1	0.1849625698	1	-0.199218763	7	-121.1420298
8	1	0.1849625698	1	-0.199218763	8	-120.0690294
9	1	0.1849625698	1	-0.199218763	9	-121.5735748
10	1	0.1849625698	1	-0.199218763	10	-122.1627516
11	1	0.1849625698	2	-1.561308219	1	4.9070152312
12	1	0.1849625698	2	-1.561308219	2	4.4006148461
13	1	0.1849625698	2	-1.561308219	3	5.1028257754
14	1	0.1849625698	2	-1.561308219	4	6.3511197644

图 3-8

若认为变量 a 与变量 b 仅仅是为了产生变量 c 而存在的过渡变量,那么就可以删除变量 a 与变量 b,操作方法如下所示。

```
data b;
set a;
drop a b;
run;
```

删除变量 a 和变量 b 的意思就是保留变量 i、变量 j、变量 k 和变量 c,所以也可以使用另外一种方法删除变量 a 与变量 b,操作方法如下所示。

```
data c;
set a;
keep i j k c;
run;
```

第三种方法是在产生数据的过程中,使用 keep 或 drop 指令直接删掉变量,相关操作方法如下所示。

```
data a;
do i=1 to 3;
a=ranuni(1);
do j=1 to 10;
b=rannor(1);
do k=1 to 10;
c=5+3*a+b*(-3)+rannor(2);
drop a b;
/* 或者亦可采用 keep i j k c;
也会得到相同的结果*/
output;
end;
end;
end;
run;
```

3.3 保留、删除观测值

保留、删除观测值有与保留、删除变量相应的程序，不过处理过程会较复杂，因为需要给定条件，条件的设定因使用者的需求而定，本节仅介绍一些设定条件的方法，使用者需要根据研究或所要的数据需求设定适当的条件，本节相关数据皆采用前一节最后输出的文件 a 中的数据。

```
data a1;
set a nobs=x;
if _n_/x<=1/2 then output;
run;
data a2;
set a nobs=x;
if _n_/x>1/2 then delete;
run;
```

在上述程序中，a1 采用的是保留法，a2 采用的是删除法，两者的结果都是输出相同的观测值。程序中语法含义如表 3-2 所示。

表 3-2

语 法	含 义
nobs=x	读取文件时的指令，要求 SAS 产生一个临时变量 x (可自行命名，不一定要为 x)，其数值等于文件的总观测值，此处的 x 在程序执行完毕后不会保留下来
n	SAS 内置的临时变量，表示该笔观测值为整个数据的第几个观测值，若数据有 100 笔，SAS 会对第 1 个观测值设定为 _n_=1，第 2 个观测值为 _n_=2，依次类推直到最后 1 笔数据
if ... then...	为条件句语法，常见的条件为比较句法，如变量 a 大于 (小于、等于) 变量 b，执行句则有输出、删除、设定其他变量等含义
output	输出观测值
delete	删除观测值

3.4 抽样方法

前面介绍了用户如何根据自己的目标来选取变量和观测值，但不需要将所有观测值直接拿出来进行检验，一般用随机抽样的方法较为合适。

所谓抽样的好坏，在于抽取出来的样本是否能够代表整体的特性。为了使样本足以代表整体，原则上希望被抽出的每一个体的机会相等，以免抽出的样本只具有某一种特征，而无法代表整体。

首先对抽样概念做简单介绍。抽样调查的对象为基本调查单位，在本例中为抽选出来的 200 家中国台湾上市公司。抽样调查对象的全体称为母群体，在本例中为台湾上市的股票。在具体研究中，要完全获得母群体的信息通常不太可能或存在一定的困难，因而不得不考虑能够获得的基本调查单位或更大的单位组成的集合。这种集合称为抽样母体，其中的单位称为抽样单位。当抽取出来的样本能代表母

群体时，则称此样本具有随机性，为随机样本。

SAS 针对随机抽样提供了 proc surveyselect 程序语言，可以帮助用户撰写随机抽样的语法，其中包含了简单随机抽样 (simple random sampling; SRS)、简单放回随机抽样 (unrestricted random sampling; URS)、系统随机抽样(systematic random sampling;SYS)以及连续随机抽样(sequential random sampling; SEQ)，本节介绍较常用的 SRS 和 URS。

1. 简单随机抽样（选后不放回）

简单随机抽样是指母群体中每一个样本被抽到的概率相同，相关程序操作如下所示：

```
libname aa 'D:\The Application of SAS in Financial Research\CH03\data';
data a;
set aa.random;
run;
proc surveyselect data=a
method=srs
/*方法为简单随机抽样*/
n=10
/*抽出 10 个样本*/
rep=2
/*重复进行两次*/
seed=1
/*种子设为 1 必为整数
设为 0 则每次都不固定*/
out=b;
/*将抽样结果输出产生命名为 b 文件*/
run;
```

在执行抽样语法之后，SAS 系统会出现如图 3-9 所示界面，告诉使用者相关的抽样信息，但这部分的信息对部分使用者来说可能没有任何帮助，此时可以加入 noprint 的指令，要求 SAS 不要输出结果，相关程序如下。

SAS系统 SURVEYSELECT过程	
选择方法	简单随机抽样
输入数据集	A
随机数种子	1
样本大小	10
选择概率	0.007746
抽样权重	129.1
操作次数	2
总样本大小	20
输出数据集	B

图 3-9

```
proc surveyselect data=a noprint
method=srs
n=10
rep=2
seed=1
out=b;
run;
```

而除了指定样本个数外，还可以指定抽出样本的特定比例，相关程序如下。

```
proc surveyselect data=a
method=srs
rate=0.01
/*要求抽出总样本的 0.01，也就是 1%的样本*/
rep=2
seed=1
out=c;
run;
```

上述指令介绍了两种随机抽样的方法：抽取 10 个样本的指令和抽取母群体样本 1%比例的样本，在原始的随机样本里包含了 1291 家上市公司股票的代码及其样本的前两位（产业代码），接下来，采用产业级别进行系统抽样的语法介绍，如表 3-3 所示。

表 3-3

语 法	含 义
n=10	选取 10 个样本，若是简单随机抽样，不得超过总样本数，且该指令不得与 rate 同时使用
rate=0.01	选取 1%的样本，rate=1 表示选取 100%的样本，该指令不得与 n 同用，取值介于 0~1 间
rep=2	重复抽取样本 2 次，亦即该随机抽样会执行两次，若省略表示只抽取 1 次
seed=1	起始值设为 1，可设定范围为 $1 \sim 2^{32}-1$ ，若为负值或小于 0，则以系统开启时间为默认值，取值为整数
out=b	将抽取样本的结果输出到 b
method=srs	采用的抽取样本方法为 SRS（简单随机抽样）

简单随机抽样根据使用者的需求会重复抽取多次，例如均值抽样分配就是利用简单随机抽样重复抽取所得的。

图 3-10 显示的结果 table b 抽取了两次，每次抽取 10 个样本，总共有 3 个变量：第几次抽取、公司代码及公司代码前两位，其中 sample replicate number 并非真实的变量名，是 SAS 的标号，若要查看真实变量名，选中该变量双击鼠标左键即可得到。

	Sample Replicate Number	code	ind
1	1	1436	14
2	1	1460	14
3	1	2059	20
4	1	2403	24
5	1	2903	29
6	1	3376	33
7	1	3452	34
8	1	8103	81
9	1	9103	91
10	1	9801	98
11	2	1462	14
12	2	2353	23
13	2	2424	24
14	2	2458	24
15	2	3323	33
16	2	5465	54
17	2	6203	62
18	2	6248	62
19	2	8931	89
20	2	9919	99

图 3-10

编写程序时，往往需要查看 SAS 的相关文件，如果每次看到的都是 label 的变量名称，而非实际 Name 的名称，在进行变量运算时会出现麻烦，在图 3-11 中，真实的变量名为 replicate，可以使用下列语法来进行消除操作。相关语法介绍如表 3-4 所示。

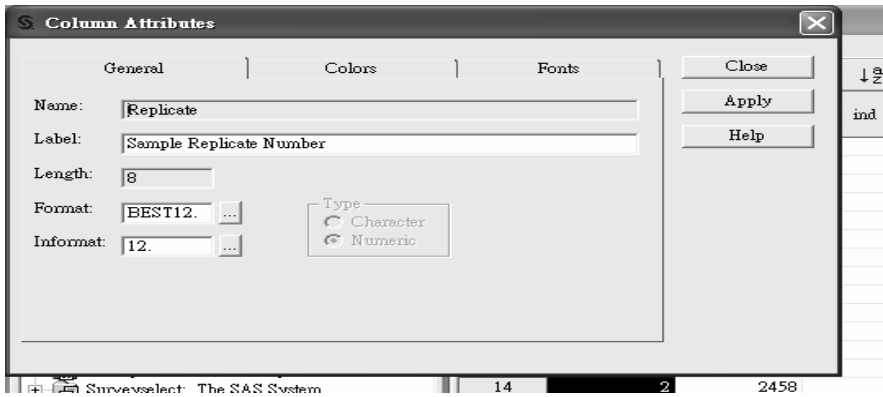


图 3-11

```
option nolabel;
```

表 3-4

语 法	含 义
option	SAS 的选项语法，有一连串的功能可以选择
nolabel	要求 SAS 移除掉 label 的功能
label	要求 SAS 执行 label 的功能

执行上述指令后，table b 的结果如图 3-12 所示。

	Replicate	code	ind
1	1	1436	14
2	1	1460	14
3	1	2059	20
4	1	2403	24
5	1	2903	29

图 3-12

2. 非限制随机抽样（重置抽样、选后放回）

简单随机抽样是将样本依序抽出，例如从 10 个样本中抽出 5 个，第 1 次抽取为 10 抽 1，第 2 次抽取为从剩下的 9 个样本中抽取 1 个，依次类推，抽取出来的样本不会重复。而在非限制随机抽样里，每次都是从 10 个样本里抽取 1 个，为了容易观察到重复样本的特性。首先用简单随机抽样方法抽取出 10 个样本作为非限制随机抽样的抽样母体，相关语法如下所示。

```
data b;
set b;
if replicate=1 then output;
keep code ind;
run;
proc surveyselect data=b
method=urs
n=15
rep=2
seed=1
out=c;
run;
proc surveyselect data=b
method=urs
rate=1
rep=2
seed=1
out=d;
run;
```

上述程序中的 method 更改为 urs，表示非限制随机抽样，抽取的样本数目可以大于总样本，非限制随机样本方法在日常生活中较少见；在学术研究中，其进阶应用范围则是拔靴法的范畴，学术研究往往以重置抽样进行，即重复抽取样本数等于总观测值的样本，该方法则可以利用 rate=1，也就是抽取 100% 的样本数目，至于重复次数则可能是千次或万次不等。

	Replicate	code	ind	NumberHits
1	1	1436	14	2
2	1	1460	14	1
3	1	2059	20	1
4	1	2403	24	1
5	1	3376	33	2
6	1	9801	98	3
7	2	1436	14	1
8	2	2059	20	3
9	2	3376	33	1
10	2	3452	34	1
11	2	9103	91	2
12	2	9801	98	2

图 3-13

table d 的结果如图 3-13，亦即抽取 100% 的样本，在第一次的样本抽取中，只抽取出 6 个不同的样本，但是其中 1436 与 3376 被重复抽取出 2 次，9801 则被重复抽取出 3 次。相关语法介绍如表 3-5 所示。

表 3-5

变 量 名 称	说 明
replicate	第几次抽取的样本
numberhits	非限制简单随机抽样中，样本被重复抽取的次数

3.5 总结

本章介绍了变量及观测值的创建及选取方法，读者可以根据自己的兴趣，创建出适合的样本，或者是针对现有的样本利用随机抽样方法选取样本，本章所介绍的抽样方法仅是其他方法的第一个步骤，有关更高级的应用方法在均值抽样分配以及拔靴法中会有更详细的介绍。

第 4 章

数据的排序、分组与转置

当我们在取得数据时，样本数据可能是混杂难以辨认的，我们需要将数据进行一些处理以掌握其规律性，此时将数据进行排序就是可行的方法，将数据依照某个变量由大到小排列或者由小到大排列。而如果我们想要将数据分组，例如财务上的小公司效应，我们就可以将公司依照市值分组，找出大公司及小公司，观察其报酬绩效是否真的是小公司优于大公司；甚至，当我们取得数据的时候，数据未必是依照观测值排列的，我们就需要将数据进行转置。

SAS 在执行程序时，通常以列为变量（规模、账面价值、股票报酬率），行为观测值（公司或个人）为佳，如果取得的数据直行是观测值，这样就需要经过一些转换方能进行之后的程序应用，接下来我们开始讲述在 SAS 中如何将数据进行排序、分组及转置。

4.1 数据的排序 (proc sort)

排序语法在进行编写程序语句中往往只是短短的一行，但也是最关键的一行语法，若日志文档出现“没有正确排序 BY 变量”或者“BY variables are not properly sorted on data set”时，就需要回过头去修正增列排序语句，以下以 example_4_1.sas 为例来介绍排序语法，example4-1 数据读取如表 4-1 所示。

表 4-1

libname aa "D:\The Application of SAS in Financial Research\CH04\data";
data a;
set aa.sort;
run;

首先读取样本数据，将文档存取为 table a，该文档包含年、月、公司代码、股票报酬率、股票周转率、股票市值 (y、m、code、ret、turn、mv)，所有数据都为月数据，若数据有缺失，变量将以 “.” 表示。我们先检查数据读取后的结果，如图 4-1 所示，再来决定如何对数据进行整理。

	y	m	code	ret	turn	mv
1	2005	1	1101	-3.83	9.16	53179
2	2005	2	1101	0.5	5.04	53444
3	2005	3	1101	-9.41	10.62	48417
4	2005	4	1101	3.83	5.97	50269
5	2005	5	1101	-6.05	5.7	47226
6	2005	6	1101	10.08	9.93	51989
7	2005	7	1101	0.76	5.61	52386
8	2005	8	1101	2.53	5.74	53709
-	----	---	..--	-----

图 4-1

在原始数据中，数据的排列顺序是依照各只股票由 1 月到 12 月的顺序排列的，若我们想要将数据进行新的排列，看看规模最小排到最大的顺序会是如何，则可以采用数据排序语法的程序来进行，如表 4-2 所示。

表 4-2

proc sort data=a;by mv;
run;

在 SAS 中进行排序的语法非常简洁，仅有短短的一行就可以处理排序，其内建的 proc sort 语法就是专门为了将数据排序用的，首先我们先观察一下程序执行的结果，数据排序结果 (tabeca) 如图 4-2 所示。

	y	m	code	ret	turn	mv
1	2005	11	3239	-57.2	.	.
2	2005	5	6249	-85.11	1.53	12
3	2005	6	1450	-29.41	8.9	13
4	2005	7	1450	0	1.22	13
5	2005	12	1224	-18.18	0.13	17
6	2005	5	1450	-63.04	1.42	18
7	2005	6	2902	-20	2.49	19
8	2005	11	1224	-21.43	0.03	20

图 4-2

由程序执行的结果发现，股票市值数据是按由小到大的升幂排序的，其中第一家公司的市值为缺失值，而小公司中，1450、1224 重复出现，因为这些公司的规模连续几个月都相当小，这使得将所有数据一起依照规模排列有点问题。接下来我们介绍如何将公司每个月都依照规模由大到小排列，降幂排序语法如表 4-3 所示；而除了要求由大到小排列之外，亦要求 SAS 将排列后的结果输出到另外一个文档中。

表 4-3

proc sort data=a out=b;by y m descending mv;
run;

首先，我们先检查一下输出的 table b（降幂排序）的结果，如图 4-3 所示。

	y	m	code	ret	turn	mv
1	2005	1	2330	2.97	4.05	1209102
2	2005	1	2412	2.4	2.1	617454
3	2005	1	2882	-3.85	2.72	519915
4	2005	1	6505	3.48	2.49	498162
5	2005	1	2317	-4.76	5.48	452343
6	2005	1	2303	-0.98	6.94	361177
7	2005	1	2002	-0.56	5.28	352046
8	2005	1	1303	-3.7	2.29	326785

图 4-3

在上述结果中，2330 为 2005 年 1 月规模最大的公司、其次为 2412、2882 等，如果要求数据由 12 月排列到 1 月，则可用表 4-4（多变量降幂排序语法）的程序来表示。

表 4-4

proc sort data=a out=b;by y descending m descending mv;
run;

在 SAS 中若要求要对数据依照某些变量进行合并、计算均值、分组、转置等的话，那么在此之前，数据都要依照这些变量进行排序才行，否则 SAS 将不进行要求执行的程序，因此要特别注意并且熟悉

排序的功能。

排序的语法亦可在删除数据时用，在我们取得样本数据后，有时候数据库来源的数据在管理上并不好，会导致可能有同一笔数据重复出现的情形，在 proc sort 中提供了删除重复数据的语法。

首先，我们先将数据复制为重复的笔数，再进行删减的动作，制作重复数值的语法如表 4-5 所示。

表 4-5

```
data a;
set a ;
do i=1 to 2;
output;
end;
drop i;
run;
```

该语法在第 3 章已经介绍过了，我们要求 SAS 将 table a 重置两次，接着将 i 数据删除。重复数值语法执行的结果如图 4-4 所示。

	y	m	code	ret	turn	mv
1	2005	11	3239	-57.2	.	.
2	2005	11	3239	-57.2	.	.
3	2005	5	6249	-85.11	1.53	12
4	2005	5	6249	-85.11	1.53	12
5	2005	6	1450	-29.41	8.9	13
6	2005	6	1450	-29.41	8.9	13
7	2005	7	1450	0	1.22	13
8	2005	7	1450	0	1.22	13

图 4-4

由图 4-4 可知，每个观测值都重复了，由这样的数据来进行数据分析，必然会有极大的问题，或者我们在进行分析的时候，不小心将部分数据重复了，就必须利用 proc sort 这个程序语法将数据还原。删除重复值语法如表 4-6 所示，删除重复值语法的日志文件如图 4-5 所示。

表 4-6

```
proc sort data=a noduprecs;by y;
/*只是要删数据的话，可以 by 任意变量*/
run;
```

```
139 proc sort data=a noduprecs;by y;
140 /*只是要删数据的话，可以by 任意变量*/
141 run;

NOTE: 从数据集 WORK.A. 读取了 18402 个观测
NOTE: 9201 个重复观测已删除。
NOTE: 数据集 WORK.A 有 9201 个观测和 6 个变量。
NOTE: "PROCEDURE SORT" 所用时间 (总处理时间):
      实际时间      0.04 秒
      CPU 时间      0.01 秒
```

图 4-5

由 SAS 的日志文件能知道执行程序后，SAS 就将 9201 笔重复的数据删除掉了，用户就可以放心地使用 9201 笔完全不重复的数据进行数据的分析，而 SAS 亦可以删除同组别的多余观测值，或者说只想保留相同月份（组别）的第一笔观测值时使用它。依照特定变量删除重复值的语句如表 4-7 所示，其执行结果如图 4-6 所示。

表 4-7

```
proc sort data=a nodupkey; by y m;
/*要求 SAS 删除掉 by 之后重复的笔数*/
run;
```

	y	m	code	ret	turn	mv
1	2005	1	6602	-71.25	0.08	27
2	2005	2	8010	-18.22	0.05	47
3	2005	3	3133	-95	0.16	22
4	2005	4	3146	-80.87	1.41	22
5	2005	5	6249	-85.11	1.53	12
6	2005	6	1450	-29.41	8.9	13
7	2005	7	1450	0	1.22	13
8	2005	8	1224	25	0.05	42
9	2005	9	1224	-42.22	0.09	24
10	2005	10	1224	7.69	0.13	26
11	2005	11	3239	-57.2	.	.
12	2005	12	1224	-18.18	0.13	17

图 4-6

执行程序删除相同月份的其他数据后，table a 只保留了 12 笔观测值的数据，因此在使用时，建议删除数据时多利用 out 的语法，不要针对原始的数据进行更改为佳。依照变量删除重复值且输出到独立文档的语句如表 4-8 所示，其输出独立文档的日志文档如图 4-7 所示。

表 4-8

```
proc sort data=a nodupkey out=a1; by y ;
run;
```

```
92  proc sort data=a nodupkey out=a1; by y ;
93  run;

NOTE: There were 12 observations read from the data set WORK.A.
NOTE: 11 observations with duplicate key values were deleted.
NOTE: The data set WORK.A1 has 1 observations and 7 variables.
NOTE: PROCEDURE SORT used (Total process time):
      real time          0.01 seconds
      cpu time           0.01 seconds
```

图 4-7

SAS 并未变更原始的 table a，而是将删除重复值的数据输出成 table a1，这样在日后进行数据的比对时，将会比较方便，并且避免犯了无法补救的错误而使得数据全部丢失。

本节重点语法如表 4-9 所示。

表 4-9

语 法	含 义
out	将数据输出到某个文档，如果不设定，则直接将原文档排序
by	依照后列变量排序，若有多个变量，例如 by y m，则会在每个 y 中排列 m
descending	为要求由大到小排列，若不撰写则会由小到大排列，一次只能执行一个变量，若有多个变量要由大到小排列，则要在由大到小排列的变量前面都要加上 descending
nodupres	要求 SAS 将重复的观测值删除
nodupkey	要求 SAS 依照 by 的变量将重复的观测值删除

4.2 数据的分组（proc rank）

在财务研究中，常会使用到投资组合或者敏感性分析，这时就需要使用分组语法来进行分析，本节延续 example_4_1 中的语法，简要介绍如何对样本依据特定变量进行分组。数据分组语句整理如表 4-10 所示。

表 4-10

<pre>data b; set b; if _n_<=20 then output; run;</pre>						
---	--	--	--	--	--	--

首先我们将数据作一个限制，为了讲解方便，只要求数据是先前文档的前 20 个观测值，在 SAS 中，新的 table b 会覆盖掉前面的 table b，完全不会有任何错误产生。分组语句的数据如图 4-8 所示。

	y	m	code	ret	turn	mv
1	2005	1	2330	2.97	4.05	1209102
2	2005	1	2412	2.4	2.1	617454
3	2005	1	2882	-3.85	2.72	519915
4	2005	1	6505	3.48	2.49	498162
5	2005	1	2317	-4.76	5.48	452343
6	2005	1	2303	-0.98	6.94	361177
7	2005	1	2002	-0.56	5.28	352046
8	2005	1	1303	-3.7	2.29	326785
9	2005	1	1326	-3.31	2.64	285257
10	2005	1	1301	1.84	4.42	282788
11	2005	1	2881	-2.15	2.27	262480
12	2005	1	2886	-3.65	4.11	239816
13	2005	1	2409	5.23	30.63	239473
14	2005	1	2357	1.78	6.92	219550
15	2005	1	2891	-5.8	3.66	206549
16	2005	1	3009	7.24	26.36	175285
17	2005	1	2382	-7.02	6.08	164684
18	2005	1	3045	-7.04	2.92	161316
19	2005	1	2863	-12.09	6.76	150831
20	2005	1	2880	-1.85	2.77	148418

图 4-8

接下来我们将数据依照 mv 变量分组，按由小到大的顺序分成 5 组，数据分组语法如表 4-11 所示。

表 4-11

option nolabel;							
proc rank data=b out=b groups=5;							
var mv;							
ranks mvrnk;							
run;							

程序先要求 SAS 不要输出变量的 label，这个指令在撰写程序过程中相当方便，最好在一开始就写上，接下来的程序就是要求将组别分成 5 组，数据分组结果如图 4-9 所示。

	y	m	code	ret	turn	mv	mvrnk
1	2005	1	2330	2.97	4.05	1209102	4
2	2005	1	2412	2.4	2.1	617454	4
3	2005	1	2882	-3.85	2.72	519915	4
4	2005	1	6505	3.48	2.49	498162	4
5	2005	1	2317	-4.76	5.48	452343	3
6	2005	1	2303	-0.98	6.94	361177	3
7	2005	1	2002	-0.56	5.28	352046	3
8	2005	1	1303	-3.7	2.29	326785	3
9	2005	1	1326	-3.31	2.64	285257	2
10	2005	1	1301	1.84	4.42	282788	2
11	2005	1	2881	-2.15	2.27	262480	2
12	2005	1	2886	-3.65	4.11	239816	2
13	2005	1	2409	5.23	30.63	239473	1
14	2005	1	2357	1.78	6.92	219550	1
15	2005	1	2891	-5.8	3.66	206549	1
16	2005	1	3009	7.24	26.36	175265	1
17	2005	1	2382	-7.02	6.08	164684	0
18	2005	1	3045	-7.04	2.92	161316	0
19	2005	1	2883	-12.09	6.76	150831	0
20	2005	1	2880	-1.85	2.77	148418	0

图 4-9

上述数据分组结果中 SAS 在分组中是由 0 开始分组的，最小的组别为 0，最大的组别为 4，刚好 5 组，如果要按由大到小的顺序分组，则可使用 descending 指令，降幂分组语法如表 4-12 所示，执行结果如图 4-10 所示。

表 4-12

proc rank data=b out=b groups=5 descending;							
var mv;							
ranks mvrnk;							
run;							

	y	m	code	ret	turn	mv	mvrnk
1	2005	1	2330	2.97	4.05	1209102	0
2	2005	1	2412	2.4	2.1	617454	0
3	2005	1	2882	-3.85	2.72	519915	0
4	2005	1	6505	3.48	2.49	498162	0
5	2005	1	2317	-4.76	5.48	452343	1
6	2005	1	2303	-0.98	6.94	361177	1
7	2005	1	2002	-0.56	5.28	352046	1
8	2005	1	1303	-3.7	2.29	326785	1
9	2005	1	1326	-3.31	2.64	285257	2
10	2005	1	1301	1.84	4.42	282788	2
11	2005	1	2881	-2.15	2.27	262480	2
12	2005	1	2886	-3.65	4.11	239816	2
13	2005	1	2409	5.23	30.63	239473	3
14	2005	1	2357	1.78	6.92	219550	3
15	2005	1	2891	-5.8	3.66	206549	3
16	2005	1	3009	7.24	26.36	175265	3
17	2005	1	2382	-7.02	6.08	164684	4
18	2005	1	3045	-7.04	2.92	161316	4
19	2005	1	2883	-12.09	6.76	150831	4
20	2005	1	2880	-1.85	2.77	148418	4

图 4-10

这样，SAS 就会将文档 tableub 保存成按由大到小的顺序排列的数据，不但是文档取代掉了，连变量也顺带取代掉了，那么 SAS 是否只能针对 1 个变量分组，而不能针对多个变量分组呢？这个问题牵涉两种做法，独立分组与相依分组，前者是仅看这些变量在观测值的大小区分组别，后者是新的变量根据旧有的变量再区分出新的组合，下面我们分别介绍如表 4-13 所示的多变量独立分组语法，其执行结果如图 4-11 所示。

表 4-13

```

data b;
set b;
drop y m ;
run;
proc rank data=b out=b groups=5 ;
var mv turn ret;
ranks mvrnk turnrank retnrk;
run;
```

我们先将数据再整理一下，在上述程序中数据的年份、月份的意义并不大，所以我们先删除之，在这个程序中，要求 SAS 对三个变量（ret, turn, mr）进行分组。

	code	ret	turn	mv	mvrnk	turnrnk	retrnk
1	2330	5.04	11.1	1545626	4	3	2
2	2317	6.19	5.29	736959	4	2	3
3	2412	-0.35	1.5	547990	4	0	1
4	6505	-4.59	2.01	522029	4	0	0
5	2882	-1.98	2.77	507191	3	0	1
6	2303	-0.53	13.03	368181	3	3	1
7	1303	12.02	3.86	334549	3	1	4
8	2454	10.41	27.45	334387	3	4	3
9	2357	6.32	8.45	295376	2	2	3
10	2409	2.08	20.69	285696	2	4	2
11	1326	0.38	2.97	284282	2	0	2
12	1301	0.4	3.1	280469	2	1	2
13	2002	-0.4	8.71	263648	1	2	1
14	3009	19.26	24.9	248915	1	4	4
15	2886	-2.95	5.15	242658	1	1	0
16	2881	5.42	4.14	227441	1	1	3
17	2498	23.69	63.04	219921	0	4	4
18	2353	11.04	13.25	185997	0	3	4
19	2891	-3.52	12.75	184359	0	3	0
20	2382	-10.31	6.21	149332	0	2	0

图 4-11

由图 4-11 所示的多变量独立分组执行结果中可以发现，SAS 分别对 ret、turn 和 mv 三个变量进行了分组，其分组方式分别是依照各个观测值的 ret、turn 和 mv 在所有观测值的大小分组的，如果我们想要 ret 以及 turn 是根据在 mvrnk 里面的数据排序分组，亦即每个 mvrnk 里的数据都分成 4 组（因为数据只有 4 笔），则可以用表 4-14 所示的相依分组语法的程序来进行。

表 4-14

```

proc sort data=b;by mvrnk;
run;
proc rank data=b out=c groups=4;
var    turn ret;
ranks turnrnk retrnk;
by mvrnk;
run;

```

在相依分组语法程序中，由于我们要根据规模的组合来进行新的分组，因此需要先将样本数据依照 mvrnk 排序，接着再利用该数据进行分组。在程序中增加了 by 这个指令，亦即要求 SAS 将数据依照在每个 mvrnk 中进行分组，该程序的运行结果就是相依分组，相依分组语法执行结果如图 4-12 所示，和独立分组不一样的是，相依分组会因为分组的先后顺序的不同，而使得结果改变，例如此处先以 mv 进行分组，之后再对 turn 分组；如果顺序颠倒，先针对 turn 分组，再进行 mv 的分组，股票成分的组合就会改变，而独立分组则是互相不会干扰的。

	code	ret	turn	mv	mvrnk	turnrnk	retrnk
1	2498	23.69	63.04	219921	0	3	3
2	2353	11.04	13.25	185997	0	2	2
3	2891	-3.52	12.75	184359	0	1	1
4	2382	-10.31	6.21	149332	0	0	0
5	2002	-0.4	8.71	263648	1	2	1
6	3009	19.26	24.9	248915	1	3	3
7	2886	-2.95	5.15	242658	1	1	0
8	2881	5.42	4.14	227441	1	0	2
9	2357	6.32	8.45	295376	2	2	3
10	2409	2.08	20.69	285696	2	3	2
11	1326	0.38	2.97	284282	2	0	0
12	1301	0.4	3.1	280469	2	1	1
13	2882	-1.98	2.77	507191	3	0	0
14	2303	-0.53	13.03	368181	3	2	1
15	1303	12.02	3.86	334549	3	1	3
16	2454	10.41	27.45	334387	3	3	2
17	2330	5.04	11.1	1545626	4	3	2
18	2317	6.19	5.29	736959	4	2	3
19	2412	-0.35	1.5	547990	4	0	1
20	6505	-4.59	2.01	522029	4	1	0

图 4-12

分组语法介绍如表 4-15 所示。

表 4-15

语 法	含 义
out	将分组后的结果输出到某个 table
groups	要求将数据分为几组
descending	将数据由大到小分组，SAS 分组为 0 开始
var	在 proc rank 中是指针对哪些变量分组
ranks	分组的变量名称命名
by	要求 SAS 在哪些变量内进行分组，在使用 by 这个指令之前，数据必须要先用这些变量排序，且这些变量最好为间断变量

4.3 数据的转置

一般而言，在针对原始数据的整理中，不会使用转置语法，但利用一些实证模型所得到的结果，若要整理，常常会使用转置语法，将其整理成论文报告可以使用的表格，而且在一些特定情况下，我们将数据转置，可以在检验数据方面更为方便，并且更能清楚了解数据的特性，本节采用 example_4_2.sas 的例子来介绍相关的转置语法。数据抽样的语法如表 4-16 所示。

表 4-16

```
libname aa "D:\The Application of SAS in Financial Research\CH04\data";  
data a;  
set aa.random;  
  
run;  
  
proc surveyselect data=a out=b noprint  
method=srs  
seed=1  
rep=5  
n=5;  
run;
```

本节先使用第 3 章所介绍的简单随机抽样程序抽取 5 次样本，每次抽取 5 家公司，其结果会是如何呢？数据抽样结果如图 4-13 所示。

	Sample Replicate Number	code	ind
1	1	2060	20
2	1	2404	24
3	1	2905	29
4	1	8110	81
5	1	9904	99
6	2	1436	14
7	2	1460	14
8	2	3376	33
9	2	3452	34
10	2	9801	98
11	3	1462	14
12	3	3325	33
13	3	6207	62
14	3	6261	62
15	3	8936	89
16	4	2353	23
17	4	2424	24
18	4	2458	24
19	4	5465	54
20	4	9919	99
21	5	2444	24
22	5	3009	30
23	5	3152	31
24	5	3508	35
25	5	5455	54

图 4-13

由图 4-13 可见，看起来结果似乎很清楚，但事实上我们在阅读上仍有所不便，第 1 次抽取的样本为 2060、2404、2905、8110、9904，由于是在竖列上摆置，我们不容易说明这是第 1 次抽取出来的样本，而利用如表 4-17 所示的数据转置语法程序进行转置，再来看看如图 4-14 所示的数据转置结果是否更佳。

表 4-17

```
proc transpose data=b out=c;
var code;
by replicate;
run;
```

	Sample Replicate Number	NAME OF FORMER VARIABLE	COL1	COL2	COL3	COL4	COL5
1	1	code	2060	2404	2905	8110	9904
2	2	code	1436	1460	3376	3452	9801
3	3	code	1462	3325	6207	6261	8936
4	4	code	2353	2424	2458	5465	9919
5	5	code	2444	3009	3152	3508	5455

图 4-14

由如图 4-14 所示的数据转置结果显示，每个观测值都具有 5 个抽样样本的公司代码，看起来比较简洁一点，但唯一的缺点就是变量名变为 COL1、COL2、COL3、COL4、COL5，在变量命名上感觉不出有任何意义,但这样的好处是SAS 对于连续性的变量往往可以用COL1 ~ COL5来表示COL1 COL2 COL3 COL4 COL5 这一连串的意义，在撰写程序上反而又是一个极佳的优点。但是这是由于 COL1 到 COL5 都是代表相同的抽样样本的股票代码，所以即使变量不进行命名，我们仍可理解，若是如图 4-15 所示的股市数据读取状况，则必需另外思考，应采用如表 4-18 所示的股市数据读取与转置语法。

表 4-18

```
data a;
set aa.transpose;
run;
proc transpose data=a out=c;
by code;
run;
```

首先我们读取 transpose 这样的文档，读取的股市数据如图 4-15 所示。

	code	variable	data
1	1101	ret	0.5
2	1101	mv	200
3	1101	bm	0.8
4	1101	value	10
5	1102	ret	6.1
6	1102	mv	150
7	1102	bm	1.1
8	1102	value	8
9	2002	ret	0.42
10	2002	mv	800
11	2002	bm	0.7
12	2002	value	40

图 4-15

在该数据中共有 12 笔观测值，分别为三家公司的 ret、mv、bm 和 value 四个变量的数据，经过转置后结果会变成什么样子呢？股市数据的转置结果如图 4-16 所示。

	code	NAME OF FORMER VARIABLE	LABEL OF FORMER VARIABLE	COL1	COL2	COL3	COL4
1	1101	data	data	0.5	200	0.8	10
2	1102	data	data	6.1	150	1.1	8
3	2002	data	data	0.42	800	0.7	40

图 4-16

首先我们将数据依照公司代码进行转置，发现如图 4-16 所示的股市数据转置结果的观测值只有 3 笔，却没有任何 ret、mv、bm、value 变量的出现，为了让这些变量出现，改用如表 4-19 所示的针对多个特定变量转置语法的新程序来进行转置。多个变量转置结果如图 4-17 所示。

表 4-19

```
proc transpose data=a out=c;
var variable data;
by code;
run;
```

	code	NAME OF FORMER VARIABLE	LABEL OF FORMER VARIABLE	COL1	COL2	COL3	COL4
1	1101	variable	variable	ret	mv	bm	value
2	1101	data	data	0.5	200	0.8	10
3	1102	variable	variable	ret	mv	bm	value
4	1102	data	data	6.1	150	1.1	8
5	2002	variable	variable	ret	mv	bm	value
6	2002	data	data	0.42	800	0.7	40

图 4-17

如图 4-17 所示，采用新的转置语法之后，variable 确实都出现了，但是结果反而更加难以分辨，有没有办法让 COL1 编码为 ret、COL2 编码为 mv 这样的情况呢？可采用如表 4-20 所示的转置变量命名语法来做。转置变量命名语法的执行结果如图 4-18 所示。

表 4-20

```
proc transpose data=a out=c;
id variable;
by code;
run;
```

	code	NAME OF FORMER VARIABLE	LABEL OF FORMER VARIABLE	ret	mv	bm	value
1	1101	data	data	0.5	200	0.8	10
2	1102	data	data	6.1	150	1.1	8
3	2002	data	data	0.42	800	0.7	40

图 4-18

由表 4-20 可见，加上 id 这个指令，就能顺利地将变量名改为我们所需要的名称。但是为了更加简洁，还可以采用如表 4-21 所示的转置且结合删除语法的程序来做转置。

表 4-21

```
proc transpose data=a out=c(drop=_name_ _label_);
id variable;
by code;
run;
```

转置且结合删除语法的执行结果如图 4-19 所示。

	code	ret	mv	bm	value
1	1101	0.5	200	0.8	10
2	1102	6.1	150	1.1	8
3	2002	0.42	800	0.7	40

图 4-19

这样，我们就顺利完成了数据转置的整理。但一般在取得数据时，通常都是以观测值为列数据，变量数据为行数据的，所以该转置程序语法仅作为程序介绍所用，以帮助读者在遇到这些少数的状况时，可以快速地进行数据整理转置。

转置语法介绍如 4-22 所示。

表 4-22

语 法	含 义
out	将转置后的结果输出到某个 table
var	针对哪些变量进行转置， 若省略该变量，SAS 会针对所有非文本变量进行转置
by	要求 SAS 在哪些变量内进行转置，在使用 by 这个指令之前，数据必须要先利用这些变量排序，且这些变量最好为间断变量
id	要求转置变量命名

4.4 总结

本章介绍了三个重要的程序，分别是排序、分组以及转置，其中尤其以排序的程序最为重要，其语法虽然简短，但是在整理数据时，只要是跟依照某个变量分组、转置或者之后的合并、计算均值、进行回归等有关，就一定要先进行排序。而该语法也因为简短，初学者在撰写程序时，反而会发生忘了要先排序，或者误以为之前已经进行了排序、现在就不需要排序的情况，因此在撰写有关 by 的指令时，请先注意 proc sort 这个程序执行了没有。

第 5 章

数据的合并

在数据的取得中，研究者或者使用者往往不能够一次性取得所有的数据，例如，对于金融业而言，其每天都会产生相同类型的数据，亦即时间序列型态的数据，此时若要同时整理，就必须先要将数据进行垂直合并，然后再进行后续分析；而也有可能需要将不同来源的数据进行合并，例如，一家公司每年的股票报酬率、分析师评级以及财务报表的数据，这些数据是在相同的时间点下，依照公司横截面上的数据进行合并的。

特别需要注意的是，在进行时间上或者垂直合并数据时，往往要求所有文件的数据栏位一致，不能是 2015 年有 A、B、C 三个变量，但在 2016 年时变量却是 A、B、D，而在进行水平合并时，也须注意是否有任意两个文件有相同变量却具有不同含义的情况，例如，CAR 在事件研究法中指的是累计异常报酬率（Cumulative Abnormal Return），但在银行业中指的是资本充足率（Capital Adequacy Ratio），在进行合并后，SAS 会根据撰写的顺序仅保留其中的一个变量，因此数据合并的语法在撰写时需特别谨慎。

先前的部分为单个文档的处理，我们在处理数据的时候，数据有时候会来自于不同的文档，要进行数据分析的话，又必须要将不同的文档合并为一个。事实上文档合并是最容易出错的，然而在 SAS 中，合并时往往不出错，这是怎么回事呢？这是因为它是一个最容易出错却又在程序执行上不会出错的处理。所以读者一定要弄清楚自己在合并文档时到底有没有出错，因为合并数据是分析数据的初步动作，如果在一开始犯的 error 没有发现，那么在最后发现时一定会感到非常煎熬，所以在合并数据的时候，绝对不能草率行事，一定要做到正确无误。为了进行程序的介绍，利用随机数函数先产生三个 table，以便于之后的讲解。产生随机乱数的语句如表 5-1 所示。乱数产生的结果如图 5-1 所示。

表 5-1

```
data a1;
do i=1 to 5;
a=rannor(1);
b=ranuni(1);
output;
end;
run;

data a2;
do i=6 to 10;
a=rannor(2);
b=ranuni(2);
output;
end;
run;

data a3;
do i=6 to 10;
a=rannor(2);
c=ranuni(2);
output;
end;
run;
```

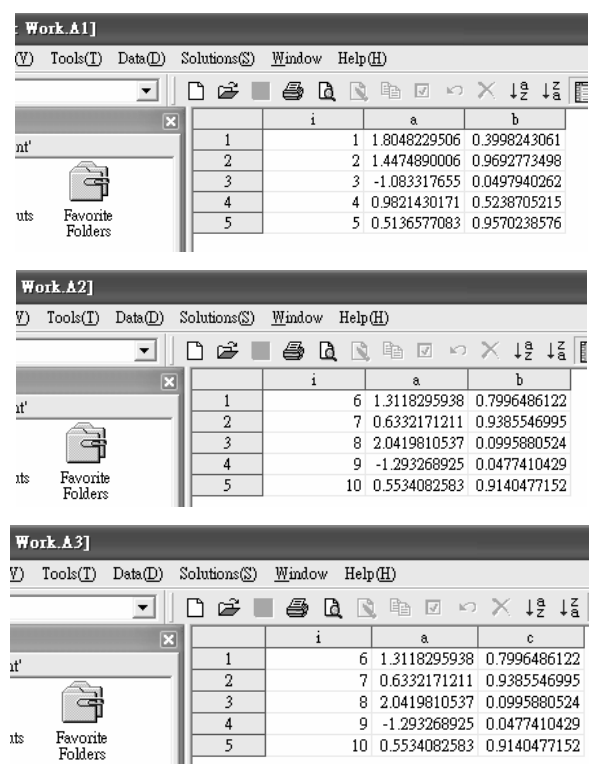


图 5-1

由图 5-1 可见，a1 有三个变量 i、a、b，a2 亦有相同的三个变量，a3 的变量 b 则改为变量 c，如果仔细看，a2 与 a3 的数据可以说是一模一样的，下面介绍如何将这几个文档合并在一起。

5.1 垂直合并

在构建研究数据时，常常需要处理跨年度的数据，而这些数据往往会因为法规要求而减少一些数据列，或者增列一些数据列，亦可能在部分年度的数据中有甲跟乙两个列，但是在其他年度则有乙跟丙两个列，这时候在合并文档时就会产生一些问题，以下采用 example_5_1.sas 的例子来介绍如何合并文档。垂直合并两个文档的语句如表 5-2 所示。

表 5-2

<pre>data a; set a1 a2; run;</pre>
--

我们先将 a1 以及 a2 合在一起，由于 SAS 的 set 是读取的指令，a 文档就会读取 a1 以后继续读取 a2，垂直合并两个文档的结果如图 5-2 所示。

	i	a	b
1	1	1.8048229506	0.3998243061
2	2	1.4474890006	0.9692773498
3	3	-1.083317655	0.0497940262
4	4	0.9821430171	0.5238705215
5	5	0.5136577083	0.9570238576
6	6	1.3118295938	0.7996486122
7	7	0.6332171211	0.9385546995
8	8	2.0419810537	0.0995880524
9	9	-1.293268925	0.0477410429
10	10	0.5534082583	0.9140477152

图 5-2

如果两个文档有不同的变量时，其垂直合并的语句如表 5-3 所示。

表 5-3

```
data b;  
set a1 a3;  
run;  
data c;  
set a2 a3;  
run;
```

接下来，我们将 a1、a3 读取为 b，a2、a3 读取为 c，观察一下合并数据的结果，如图 5-3 所示。

	i	a	b	c
1	1	1.8048229506	0.3998243061	.
2	2	1.4474890006	0.9692773498	.
3	3	-1.083317655	0.0497940262	.
4	4	0.9821430171	0.5238705215	.
5	5	0.5136577083	0.9570238576	.
6	6	1.3118295938	.	0.7996486122
7	7	0.6332171211	.	0.9385546995
8	8	2.0419810537	.	0.0995880524
9	9	-1.293268925	.	0.0477410429
10	10	0.5534082583	.	0.9140477152

	i	a	b	c
1	6	1.3118295938	0.7996486122	.
2	7	0.6332171211	0.9385546995	.
3	8	2.0419810537	0.0995880524	.
4	9	-1.293268925	0.0477410429	.
5	10	0.5534082583	0.9140477152	.
6	6	1.3118295938	.	0.7996486122
7	7	0.6332171211	.	0.9385546995
8	8	2.0419810537	.	0.0995880524
9	9	-1.293268925	.	0.0477410429
10	10	0.5534082583	.	0.9140477152

图 5-3

由图 5-3 所示的合并结果来看，SAS 会将数据上下合并，而如果文档中不存在该变量，SAS 就会自动补上，代表该数据是遗漏值。如果仔细看，在 table c 中，i 都是 6~10，我们是否能让 SAS 依照 i 合并，使得数据是 5 笔观测值，而且同时没有缺失值呢，聪明的读者可能想到有 by 这个功能，以下我们来使用 by 这个功能，看看结果会是如何？在合并语法上使用 by 的语句如表 5-4 所示。用 by 语句排序的效果如图 5-4 所示。

表 5-4

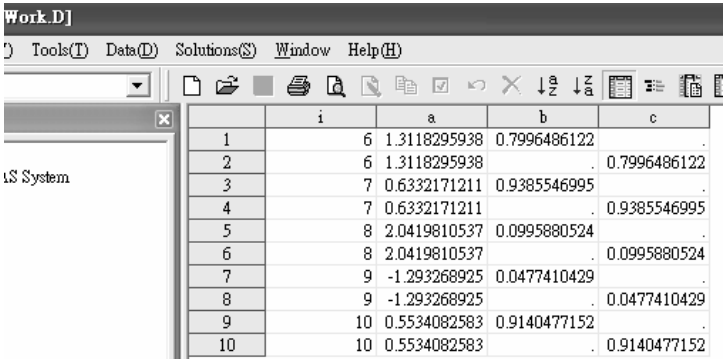
<pre>data d; set a2 a3;by i; run;</pre>				
				

图 5-4

很可惜，使用 by 这个功能，仅仅是将输出的结果依照 i 排序，并没有达到预期的目的，在 5.2 节中，我们会继续介绍这种水平合并的方法，到那时就能达成我们想要的目标了。
合并语句语法的介绍如表 5-5 所示。

表 5-5

语 法	含 义
set	读取 SAS 文档，但若有多多个文档，则具有垂直合并的意义
by	在 set 之后加上 by，则要求 SAS 将数据进行排序。但若要求 SAS 依照 by 的变量进行指令，则需要其他的附加语法

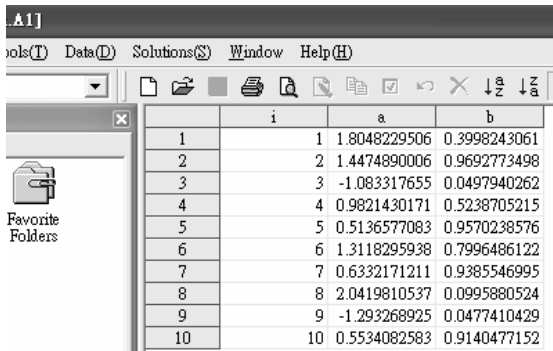
因为 set 语法是要求 SAS 读取文档的意思，SAS 会将多个文档都读到 data 的文档中，当文档极大的时候会导致合并时的计算机处理时间较长，因此在合并一个大文档和一个小文档时，可以采用 SAS 所提供的 proc append 这个附加程序步¹，如表 5-6 所示。

1 对于已经拥有庞大数据的用户而言，在垂直合并数据时使用 proc append 这个语法可以节省很多的时间。

表 5-6

```
proc append data=a2 base=a1;  
run;
```

上述程序步是基本的附加语法，其中想要附加的文档为 a2，要求 SAS 将数据附加到 a1 这个文档之下。附加程序步的执行结果如图 5-5 所示。



	i	a	b
1	1	1.8048229506	0.3998243061
2	2	1.4474890006	0.9692773498
3	3	-1.083317655	0.0497940262
4	4	0.9821430171	0.5238705215
5	5	0.5136577083	0.9570238576
6	6	1.3118295938	0.7996486122
7	7	0.6332171211	0.9385546995
8	8	2.0419810537	0.0995880524
9	9	-1.293268925	0.0477410429
10	10	0.5534082583	0.9140477152

图 5-5

这样，a1 这个 table 就多了 a2 的数据，接下来继续将 a3 附加到 a1 中，附加程序步 2 如表 5-7 所示。

表 5-7

```
proc append data=a3 base=a1;  
run;
```

结果似乎并不顺利。由图 5-6 所示的附加语句的日志文件的记录中，发现 SAS 并不执行该语法，因为 a1 与 a3 的变量不相同，所以并不能进行附加的动作，那么 SAS 在附加文档时，是否仅能附加相同的变量的文档呢？SAS 在 proc append 这个程序中提供了简单的语法，可以将不同变量的文档附加在一起，即如表 5-8 所示的附加语句与 force 的功能。

```
NOTE: Appending WORK.A3 to WORK.A1.  
WARNING: Variable c was not found on BASE file. The variable will not be added to the BASE file.  
WARNING: Variable b was not found on DATA file.  
ERROR: No appending done because of anomalies listed above. Use FORCE option to append these files.  
NOTE: 0 observations added.  
NOTE: The data set WORK.A1 has 10 observations and 3 variables.  
NOTE: Statements not processed because of errors noted above.  
NOTE: PROCEDURE APPEND used (Total process time):  
      real time      0.01 seconds  
      cpu time       0.00 seconds  
  
NOTE: The SAS System stopped processing this step because of errors.
```

图 5-6

表 5-8

```
proc append data=a3 base=a1 force;
run;
```

采用 force 这个选项，可以要求 SAS 强制将数据附加进去，我们检验一下附加 force 后的日志文件的记载，如图 5-7 所示。

```
39 proc append data=a3 base=a1 force;
40 run;

NOTE: Appending WORK.A3 to WORK.A1.
WARNING: Variable c was not found on BASE file. The variable will not be added to the BASE file.
WARNING: Variable b was not found on DATA file.
NOTE: FORCE is specified, so dropping/truncating will occur.
NOTE: There were 5 observations read from the data set WORK.A3.
NOTE: 5 observations added.
NOTE: The data set WORK.A1 has 15 observations and 3 variables.
NOTE: PROCEDURE APPEND used (Total process time):
      real time    0.00 seconds
      cpu time     0.00 seconds
```

图 5-7

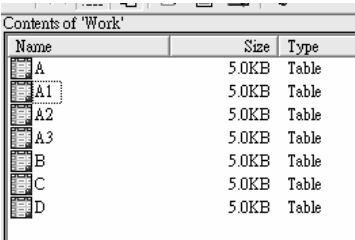
SAS 虽然提出了警告，告诉使用者两个文档的变量不同，但由于用了强制的语法，SAS 依然将数据附加了上去，但其变量依然只有 3 个，这显示 SAS 依然没有产生相关的数据。我们该如何解决这个问题呢？可以采用如表 5-9 所示的创建所有变量后执行附加程序步的方法。

表 5-9

```
data a1;
set a1;
c=.;
run;

proc append data=a3 base=a1 force;
run;
```

先重新读取 a1，建立一个都是缺失值的 c 变量，再将 a3 这个文档附加上去即可，所以在执行附加指令的时候，建议先准备一个拥有所有变量的文档再进行附加这个动作。事实上附加程序还有一个特殊的用法，即先确认，在 Work 这个逻辑库里只有 a1、a2、a3 以及 a、b、c、d 共 7 个 table，如图 5-8 所示，接下来我们撰写一个如表 5-10 所示的新的程序语法，将文档附加到不存在的新文档值的语句中。执行的日志文件如图 5-9 所示。



Name	Size	Type
A	5.0KB	Table
A1	5.0KB	Table
A2	5.0KB	Table
A3	5.0KB	Table
B	5.0KB	Table
C	5.0KB	Table
D	5.0KB	Table

图 5-8

表 5-10

proc append data=a3 base=e;	
run;	
137	proc append data=a3 base=e;
138	run;
NOTE: Appending WORK.A3 to WORK.E.	
NOTE: BASE data set does not exist. DATA file is being copied to BASE file.	
NOTE: There were 5 observations read from the data set WORK.A3.	
NOTE: The data set WORK.E has 5 observations and 3 variables.	
NOTE: PROCEDURE APPEND used (Total process time):	
real time	0.01 seconds
cpu time	0.00 seconds

图 5-9

base 文档并不存在，但是在第一步执行的时候，SAS 会将 a3 整个复制成 e 这个文档。这个语法的特性在目前似乎不是那么重要，但事实上在撰写宏程序时，往往要将执行的结果保存为一个新的文档，此时使用附加的程序语法，就可以不断地将执行结果附加到特定的文档中。在之后的介绍中，每个宏程序几乎都会用到 proc append 这个程序，读者要多加留心。

附加程序步的介绍如表 5-11 所示。

表 5-11

语法	介绍
data=	要附加的文档为 data=的文档名
base=	要被附加的文档为 base=的文档名
force	当要附加的文档在并没有所有的变量时使用，若 data 与 base 互相有对方没有的变量，则附加语法会产生问题

5.2 水平合并

在数据的水平合并中，经常会用到一些小技巧，这些技巧的程序指令其实读者在前面的章节中已经都学会了，进行水平合并的条件是当两边数据为同一笔观测值的不同变量时使用，然而，在财务研究或者其他进阶研究中，每家公司都有其产业信息，当我们要将产业信息合并的时候，也需要用到水平合并的方式，或者是将数据依照年份的 GDP、CPI 合并时，有时候也会遇到一些问题，以下采用 example_5_2.sas 的例子来介绍水平合并的 SAS 语法，如表 5-12 所示。

表 5-12

data a;	
merge a1 a2;	
run;	
data b;	

续表

```
merge a1 a3;  
  
run;  
  
data c;  
  
merge a2 a3;  
  
run;
```

和垂直合并一样，我们先将数据全部都以水平合并的方式来合并数据，以便了解水平合并指令的结果以及特性。水平合并执行的结果，如图 5-10 所示。

Figure 5-10 displays two SAS workbooks, Work.A and Work.B, showing the results of horizontal merging. Both workbooks have a menu bar (Tools, Data, Solutions, Window, Help) and a toolbar. The data is presented in a table format with columns i, a, b, and c. The rows are numbered 1 through 5.

	i	a	b
1	6	1.3118295938	0.7996486122
2	7	0.6332171211	0.9385546995
3	8	2.0419810537	0.0995880524
4	9	-1.293268925	0.0477410429
5	10	0.5534082583	0.9140477152

	i	a	b	c
1	6	1.3118295938	0.3998243061	0.7996486122
2	7	0.6332171211	0.9692773498	0.9385546995
3	8	2.0419810537	0.0497940262	0.0995880524
4	9	-1.293268925	0.5238705215	0.0477410429
5	10	0.5534082583	0.9570238576	0.9140477152

图 5-10

由 table a 以及 table b 的结果可以发现，如果两个文档里面都有共同的变量，该变量数据会被后面的文档取代掉，如 table a 的文档完全是 a2 的数据，由于 a1 与 a2 有相同的变量，导致 i、a、b 3 个变量的数据都被 a2 的数据取代掉；而 table b 中 a1、a3 共同的变量为 i 以及 a，所以在进行水平合并后 i 以及 a 的数据全部是 a3 的数据。水平合并执行结果如图 5-11 所示。

Figure 5-11 displays the SAS workbook Work.C, showing the result of horizontal merging. The table has columns i, a, b, and c. The data is the same as in Figure 5-10.

	i	a	b	c
1	6	1.3118295938	0.7996486122	0.7996486122
2	7	0.6332171211	0.9385546995	0.9385546995
3	8	2.0419810537	0.0995880524	0.0995880524
4	9	-1.293268925	0.0477410429	0.0477410429
5	10	0.5534082583	0.9140477152	0.9140477152

图 5-11

文档 c 完全没有任何问题，因为其共同的变量的数据本来就是一模一样的，那么如果我们采用 by 这个指令，结果又会变成怎么样呢？先思考一下 by 的功能，要求两个文档依照 by 的变量合并，由于两个文档没有任何共同的 i，其结果应该会与垂直合并一样。接下来，让我们介绍另外一种情况，即当两个文档的样本数不一样时的处理状况。乱数产生数据语法，如表 5-13 所示。

表 5-13

data a;
do i=1 to 5;
do j=1 to 5;
a=rannor(1);
output;
end;
end;
run;
data b;
do i=1 to 5;
b=rannor(2);
output;
end;
run;
data c;
c=ranuni(3);
output;
run;

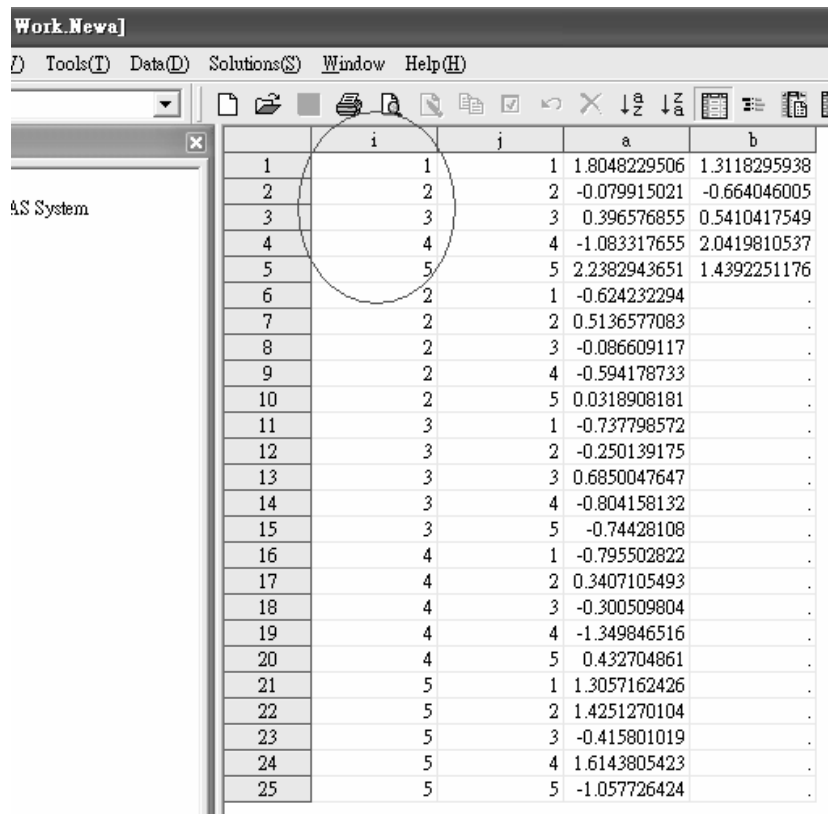
由随机数函数产生了三个文档，table a 为 25 笔数据，table b 为 5 笔数据，table c 为 1 笔数据，其中文档 a 与文档 b 的共同变量为 i，文档 c 与其他两个文档没有共同的变量，现在的目标是产生一个 table，只要有相同的 i，其变量 b 的数据都一样，且有相同的 c。

首先，我们先简单进行数据的合并，然后看程序执行的结果才有办法针对下一步的程序进行一些重要的修正。在此处虽然使用简单的例子，但是在实际操作上，我们经常需要将两种观测值不一样的数据合并，并且希望相同的年份都有完整的年度数据，相同的国家都有完整的国家数据，这是 SAS 用户都必须遵守的做法，因此读者务必了解透彻以免撰写的程序出错。

简单水平合并数据集的语句如表 5-14 所示，合并数据的结果如图 5-12 所示。

表 5-14

data newa;
merge a b;
run;



	i	j	a	b
1	1	1	1.8048229506	1.3118295938
2	2	2	-0.079915021	-0.664046005
3	3	3	0.396576855	0.5410417549
4	4	4	-1.083317655	2.0419810537
5	5	5	2.2382943651	1.4392251176
6	2	1	-0.624232294	.
7	2	2	0.5136577083	.
8	2	3	-0.086609117	.
9	2	4	-0.594178733	.
10	2	5	0.0318908181	.
11	3	1	-0.737798572	.
12	3	2	-0.250139175	.
13	3	3	0.6850047647	.
14	3	4	-0.804158132	.
15	3	5	-0.74428108	.
16	4	1	-0.795502822	.
17	4	2	0.3407105493	.
18	4	3	-0.300509804	.
19	4	4	-1.349846516	.
20	4	5	0.432704861	.
21	5	1	1.3057162426	.
22	5	2	1.4251270104	.
23	5	3	-0.415801019	.
24	5	4	1.6143805423	.
25	5	5	-1.057726424	.

图 5-12

在简单水平合并之下，结果不如想象中的顺利，SAS 只是简单地将数据水平放在了一起，且由于合并了 table b 之后，i 变量的顺序也被更改了，如果利用 by 这个指令，是否可以将结果变成我们所预期的状况呢？依变量合并数据集的语法如表 5-15 所示，其合并数据集的结果如图 5-13 所示。

表 5-15

```
proc sort data=a;by i;
run;
proc sort data=b;by i;
run;
data newa;
merge a b;by i;
run;
```

由于要先依照 i 进行合并，所以对于文档 a 与文档 b 要先依照 i 做排序的动作，接着我们再依照 i 来进行合并。

	j	i	a	b
1	1	1	1.8048229506	1.3118295938
2	2	1	-0.624232294	1.3118295938
3	3	1	-0.737798572	1.3118295938
4	4	1	-0.795502822	1.3118295938
5	5	1	1.3057162426	1.3118295938
6	1	2	-0.079915021	-0.664046005
7	2	2	0.5136577083	-0.664046005
8	3	2	-0.250139175	-0.664046005
9	4	2	0.3407105493	-0.664046005
10	5	2	1.4251270104	-0.664046005
11	1	3	0.396576855	0.5410417549
12	2	3	-0.086609117	0.5410417549
13	3	3	0.6850047647	0.5410417549
14	4	3	-0.300509804	0.5410417549
15	5	3	-0.415801019	0.5410417549
16	1	4	-1.083317655	2.0419810537
17	2	4	-0.594178733	2.0419810537
18	3	4	-0.804158132	2.0419810537
19	4	4	-1.349846516	2.0419810537
20	5	4	1.6143805423	2.0419810537
21	1	5	2.2382943651	1.4392251176
22	2	5	0.0318908181	1.4392251176
23	3	5	-0.74428108	1.4392251176
24	4	5	0.432704861	1.4392251176
25	5	5	-1.057726424	1.4392251176

图 5-13

依变量合并数据集的结果符合我们的预期,newa 的数据变得比较完善一点,首先,我们先观察 table c,单一观测值的文档如图 5-14 所示。



Work.C1	
	c
1	0.5548877095

图 5-14

c 只有一个观测值,一个变量,如果要进行合并,在简单合并下也仅会只有第一笔观测值会出现数据,其余的数据都是缺失值,那么该怎么办呢? SAS 初学者往往在此处会遇到困难,事实上有两个解决的方案,首先我们先介绍使用 by 变量合并的方法。

读者一定会觉得一头雾水,文档 c 只有一个变量一个观测值,且该变量根本与 newa 无关,如何使用 by 变量合并呢? 答案是,看不见但是依旧存在变量。既然没有,我们就当作这个变量在两个文档里都一样就可以了。

建立共同变量并且合并的语句如表 5-16 所示。

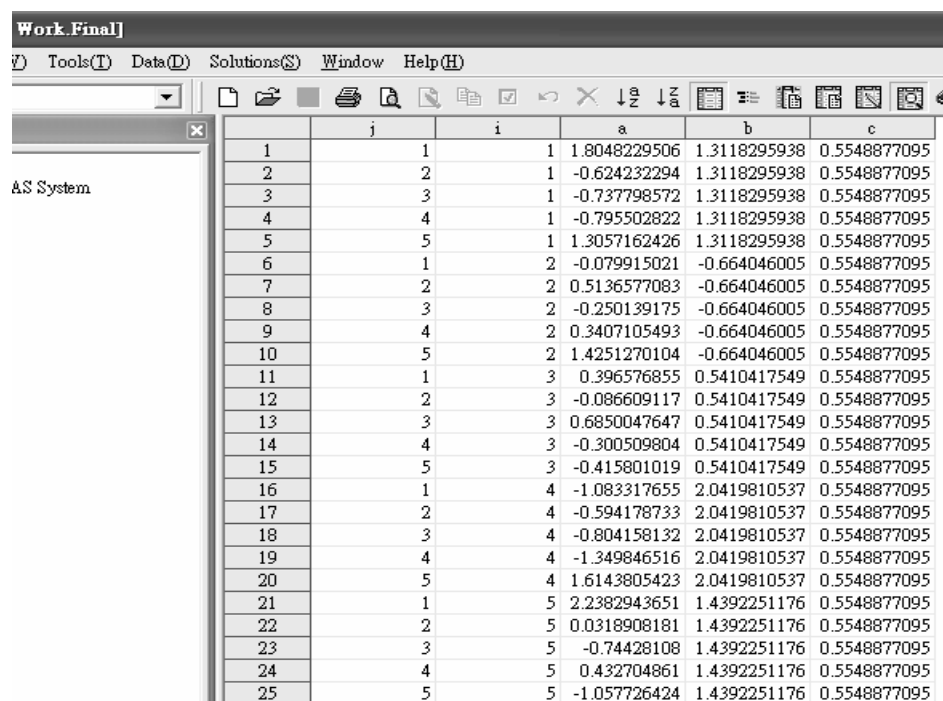
表 5-16

```

data newa;
set newa;
k=1;
run;
data c;
set c;
k=1;
run;
data final;
merge newa c;by k;
drop k;
run;

```

首先我们先针对 newa 进行处理，重新做一个变量 k，令其等于 1；接着针对 c 做处理，也令一个变量 k 等于 1，这样一来两个文档都有一个一样的 k 变量，这样就可以依照相同的变量进行合并了，而设变量这个指令，相信读者早就已经了如指掌了，接下来就可以看看合并单一数据结果的文档了，如图 5-15 所示。



	j	i	a	b	c
1	1	1	1.8048229506	1.3118295938	0.5548877095
2	2	1	-0.624232294	1.3118295938	0.5548877095
3	3	1	-0.737798572	1.3118295938	0.5548877095
4	4	1	-0.795502822	1.3118295938	0.5548877095
5	5	1	1.3057162426	1.3118295938	0.5548877095
6	1	2	-0.079915021	-0.664046005	0.5548877095
7	2	2	0.5136577083	-0.664046005	0.5548877095
8	3	2	-0.250139175	-0.664046005	0.5548877095
9	4	2	0.3407105493	-0.664046005	0.5548877095
10	5	2	1.4251270104	-0.664046005	0.5548877095
11	1	3	0.396576855	0.5410417549	0.5548877095
12	2	3	-0.086609117	0.5410417549	0.5548877095
13	3	3	0.6850047647	0.5410417549	0.5548877095
14	4	3	-0.300509804	0.5410417549	0.5548877095
15	5	3	-0.415801019	0.5410417549	0.5548877095
16	1	4	-1.083317655	2.0419810537	0.5548877095
17	2	4	-0.594178733	2.0419810537	0.5548877095
18	3	4	-0.804158132	2.0419810537	0.5548877095
19	4	4	-1.349846516	2.0419810537	0.5548877095
20	5	4	1.6143805423	2.0419810537	0.5548877095
21	1	5	2.2382943651	1.4392251176	0.5548877095
22	2	5	0.0318908181	1.4392251176	0.5548877095
23	3	5	-0.74428108	1.4392251176	0.5548877095
24	4	5	0.432704861	1.4392251176	0.5548877095
25	5	5	-1.057726424	1.4392251176	0.5548877095

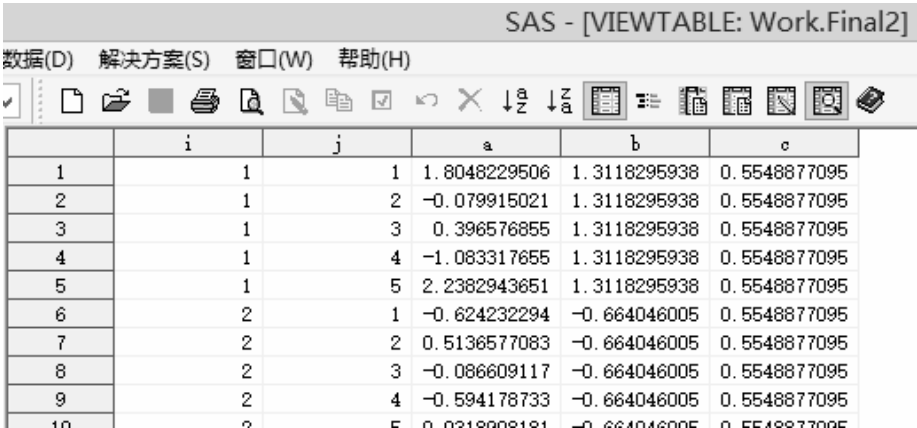
图 5-15

另外一个解决方法，在处理数据上也非常好用，我们先检验该程序语法。采用 set 合并单一数据语法如表 5-17 所示。

表 5-17

```
data newa;
set newa;
drop k;
run;
data c;
set c;
drop k;
run;
data final2;
  set newa;
  /*revise 20150803*/
  if _n_=1then set c;
run;
```

这个程序的语法是要先进行处理，先逆向操作把 k 变量删掉，接下来进行 newa 与 c 的合并，我们使用了 if _n_=1 then set c 的语法进行合并，采用 set 合并单一数据的结果如图 5-16 所示。



	i	j	a	b	c
1	1	1	1.8048229506	1.3118295938	0.5548877095
2	1	2	-0.079915021	1.3118295938	0.5548877095
3	1	3	0.396576855	1.3118295938	0.5548877095
4	1	4	-1.083317655	1.3118295938	0.5548877095
5	1	5	2.2382943651	1.3118295938	0.5548877095
6	2	1	-0.624232294	-0.664046005	0.5548877095
7	2	2	0.5136577083	-0.664046005	0.5548877095
8	2	3	-0.086609117	-0.664046005	0.5548877095
9	2	4	-0.594178733	-0.664046005	0.5548877095
10	2	5	0.0218008181	-0.664046005	0.5548877095

图 5-16

以该语句而言，表示当观测值为 1 时，开始读取 c 这个文档，由于只有一笔观测值，就会反复读取到最后一笔数值，如果读者将其改为 if _n_=2，则会由第 2 个观测值开始读取。

水平合并语法的介绍如表 5-18 所示。

表 5-18

语 法	含 义
merge	为合并两个 table 以上的指令
by	在 merge 底下，为要求依照特定变量进行合并的动作，最常被忽略的就是要先依照要合并的变量进行排序
retain	保留变量的数据给下一个观测值使用

5.3 总结

本章介绍了将不同文档合并在一起的程序语法，在分析数据的时候往往需要用到不同种类的数据，例如年度数据、月份数据、公司数据、集团数据、产业数据等，这些数据往往都是存储在不同的文档里的，重要的是如何将其合并在一起，并且做出最佳的数据呈现方式。若是在一开始整理数据就出错，会对整个分析产生极大的问题，因此 SAS 的初学者应该熟读本章的基本概念，并且经常加以复习，才能进行进一步的数据分析。

第 6 章

SAS 的数据库管理

前面几章介绍了处理 SAS 文档的语法，本章介绍如何处理逻辑库的方法，在数据分析的过程中，或多或少都会产生一些过渡而不需要的临时文档，这些文档的存在往往会影响电脑的运行速度，本章节介绍两个重要的程序，可以帮助你更好地管理文档。

6.1 文档的复制、删除与保留 (proc datasets)

在一般视窗处理的系统下，我们要复制文档有两种方法，一种是开启该文档，然后在工具选单下，选取另存新文档，就可以把该文档复制到另外的文件夹下，但也可以使用鼠标，采用拖拉式的方法直接将文档复制到别的逻辑库中，在撰写程序时，我们往往会产生许多文档，有些文档是我们需要保留的，甚至是需要输出到永久逻辑库中的，而有些部分是需要删除的，本节以 example_6_1.sas 为例来介绍该语法。

首先我们先利用随机数函数创造 a、b、c、d、e 5 个文档，这 5 个文档都只有 1 个变量以及 1 个观测值，接着我们执行如表 6-1 所示的 proc datasets 这个程序。

表 6-1

<pre>data a b c d e; a=rannor(0); output; run; proc datasets; quit;</pre>	
---	--

程序执行完毕，SAS 会在 log 窗口里呈现出结果¹如图 6-1 所示，该结果会告知使用者在 work 这个逻辑库里面有多少个文档，如果我们想要将这些文档都保存到 aa 的逻辑库中，大多都会采用如表 6-2 所示的创建文档语句的程序撰写方法。

```

5      proc datasets;                                     Directory
                                                Libref      WORK
                                                Engine      V9
Physical Name C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS Temporary Files\_TD796
File Name     C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\SAS Temporary Files\_TD796

#      Name      Member      File
      Type      Size      Last Modified

1  A      DATA      5120      2009年02月09日 16時35分23秒
2  B      DATA      5120      2009年02月09日 16時35分23秒
3  C      DATA      5120      2009年02月09日 16時35分23秒
4  D      DATA      5120      2009年02月09日 16時35分23秒
5  E      DATA      5120      2009年02月09日 16時35分23秒

6      run;

7      quit;

NOTE: PROCEDURE DATASETS used (Total process time):
      real time      0.43 seconds
      cpu time       0.01 seconds

```

图 6-1

1 由于该结果只会出现在 log 文件中，本质上该程序是无法标准化使用的，仅能针对要处理的文件夹做规划。

表 6-2

<pre>libname aa "D:\The Application of SAS in Financial Research\CH06\data\aa"; data aa.a; set a; run; data aa.b; set b; run; data aa.c; set c; run; data aa.d; set d; run; data aa.e; set e; run;</pre>
--

上述例子只有 5 个文档，但若是先前已经经过复杂的程序撰写，并且产生了上千个文档，我们不可能为这些文档一个一个地撰写程序保存成永久文档，因此 SAS 在 proc datasets 中提供了快速备份文件的语法，如表 6-3 所示。

表 6-3

<pre>libname bb "D:\The Application of SAS in Financial Research\CH06\data\bb"; proc datasets noprint; copy in=work out=bb; quit;</pre>

上述语法可以快速地将这个逻辑库里的文档都 copy 到 bb 这个逻辑库里，但是如果我们并不需要所有的文档，SAS 亦有让我们挑选文档的程序功能，同时我们也可在 datasets 的后面加上 noprint 的语法，抑制 SAS 将该语句执行结果输出，如表 6-4 所示。

表 6-4

<pre>libname cc "D:\The Application of SAS in Financial Research\CH06\data\cc"; proc datasets noprint; copy in=work out=cc; select a c e; quit;</pre>

该语法就可以只挑选出我们需要的文档转存到 cc 的逻辑库里，而在 SAS 中常常有像 drop、keep 这样成对的语法，有只挑选某些文档的语法，当然也会有排除某些文档的语法，其适用范围要看使用者的爱好以及撰写的方便，当要排除不存取的文档较少时，就可以采用如表 6-5 所示的语法。

表 6-5

```
libname dd "D:\The Application of SAS in Financial Research\data\dd";  
  
proc datasets noprint;  
copy in=work out=dd;  
exclude a c;  
  
quit;
```

以上的语法都是为将某些数据复制到别的逻辑库用的，那么是否也有管理特定逻辑库的语法呢？管理特定逻辑库的语法主要是和删除文档有关的。产生文档的程序语法，读者在第 1 章都已了解了，但有时候在执行一些程序时，有些文档的执行类型是用 proc append 产生的，这些文档基本上是会不断累加的，如果要重新执行程序，就必须在语法中增加删除文档的语法，接下来我们介绍 SAS 如何删除逻辑库里的文档，数据库程序中删除文档语法的程序如表 6-6 所示。

表 6-6

```
proc datasets library=aa noprint;  
delete a;  
  
quit;
```

如果我们不要 aa 逻辑库的 table，则可以先声明要针对那个逻辑库进行处理，接着再加 delete 的指令，就可以删除 a 了，如果是要删除掉 a 跟 b，则输入 delete a b；就可以达到目的。Delete 也有成对的指令语法，在 SAS 中亦有除了某文档以外其他都删除的语法。如 SAS 对于 bb 逻辑库的文档，就只要保留 a 与 b 两个 table。那么可采用如表 6-7 所示的数据库程序步保留特定文档的语法就可以了，处理结果如图 6-2 所示。

表 6-7

```
proc datasets library=bb noprint;  
save a b;  
  
quit;
```

```
47  proc datasets library=bb;  
                                     Directory  
Libref      BB  
Engine      V9  
Physical Name D:\SAS在财务研究上的应用\CH06\data\bb  
File Name    D:\SAS在财务研究上的应用\CH06\data\bb  
  
#  Name  Member  File  
   Type  Type    Size  Last Modified  
1  A      DATA    5120  2009年02月09日 23时32分08秒  
2  B      DATA    5120  2009年02月09日 23时32分08秒  
3  C      DATA    5120  2009年02月09日 23时32分08秒  
4  D      DATA    5120  2009年02月09日 23时32分08秒  
5  E      DATA    5120  2009年02月09日 23时32分08秒  
  
48  save a b;  
49  run;  
  
NOTE: Saving BB.A (memtype=DATA).  
NOTE: Saving BB.B (memtype=DATA).  
NOTE: Deleting BB.C (memtype=DATA).  
NOTE: Deleting BB.D (memtype=DATA).  
NOTE: Deleting BB.E (memtype=DATA).  
50  quit;
```

图 6-2

SAS 亦可将逻辑库里所有的文档全部删除，删除的语法相当简便，读者使用这个程序语法的时机有三，第一，请确认必要的文档是否已经存成永久文档；第二，在经历过长时间的使用 SAS 后，产生过多的暂存盘（存在 work 逻辑库中）导致计算机速度变慢；第三，确定要关闭 SAS 时。有时候直接关闭 SAS 会使暂存盘继续保留，该结果一样会导致计算机速度变慢，而使得网络上有传用了 SAS 之后计算机速度会变慢的流言，实际上是 SAS 害怕用户的重要数据没有保留而不删除暂存盘所致。然而如果确认执行的结果文档都已经存取完毕，则应该删除掉所有的暂存盘后关闭 SAS，这样就不会影响计算机运行的速度了¹，数据库程序步删除所有执行文档的语法如表 6-8 所示。

表 6-8

```
proc datasets kill noprint;
quit;
```

当然，该语法也可用于删除外部逻辑库的所有文档，只是使用者可以不需要在 SAS 程序中进行，可在一般窗口接口下，直接将整个数据删除即可，而利用 SAS 语法删除特定逻辑库的所有文档，可使用如表 6-9 所示的程序。

表 6-9

```
proc datasets library=cc kill;
quit;
```

使用上述语法，SAS 就会将 cc 这个声明后的逻辑库的文档全部删除，结果将在 log 窗口显示，如图 6-3 所示。

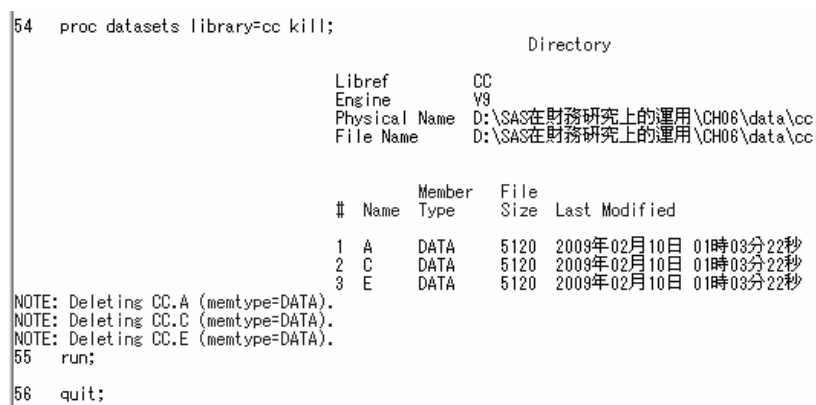


图 6-3

数据库程序步介绍如表 6-10 所示。

1 亦可选择惊叹号，选择 Terminate the SAS system 来关闭。

表 6-10

语 法	含 义
copy in=原始逻辑库 out=目标逻辑库	复制文档使用，从原始逻辑库复制，输出到目标逻辑库，若未下 select 或者 exclude 指令，会变成完全复制
select	搭配 copy 语法，选择要复制的文档
exclude	搭配 copy 语法，选择不要复制的文档
library=目标逻辑库	为程序的选项，删除文档所用，没下这个指令就会删除掉 work 逻辑库的文档
delete	选择要删除掉的文档
save	选择不要删除掉的文档
kill	为程序选项，在 proc datasets 之后撰写，删掉逻辑库所有文档的语法

最后，应该说明，本节仅介绍了 proc datasets 的复制以及删除文档的功能，proc datasets 仍有其他丰富的功能可以使用，且也可以使用 proc copy 以及 proc delete 这两个程序步来执行复制以及删除文档，读者可做进一步的研究。

6.2 结构化查询语言

结构化查询语言（Structured Query Language，SQL）是一种标准化且使用广泛的数据库查询语言，在 SAS 中提供了 proc sql 这个程序让 SAS 用户可以运用 SQL 的语法。对于用户而言，若要完全了解 SQL 的语法，可以阅读介绍 SQL 的书籍，或者学习数据库管理、数据仓储等课程。

本节仅简单介绍 SAS 中有关 SQL 的语法，并针对财务上可能用到的情况进行介绍，这部分的功能介绍在数据分析上已经足够，若要更进一步了解使用方法则非本书所能提供。读者可以将 SQL 作为 Google 这一类的网页搜索引擎，而文档为网络数据，在撰写 SQL 时，可以将我们想要的数据库全部展示出来，因为 Google 就是在搜寻我们要的数据。

而和其他的语法不同，SAS 的 SQL 语言直接是由别的系统搬移过来的，在 R 语言和 Matlab 语言中，都有相对应的 SQL 语法，因此在撰写 SQL 语法时，读者可以寻找专门介绍 SQL 语法的书籍，并不一定需要找 SAS 的教学书籍，以下采用 example_6_2.sas 作为例子来介绍，首先说明 SQL 语法的撰写步骤，如表 6-11 所示。

表 6-11

proc sql;		
create	table	tablename as
select		variable
from		sourcefile
where		condition
group	by	variable
order	by	variable
;		
quit;		

这些语法中，完全不能省略的是 select 以及 from 这两个子句（此处采用 SQL 书中的说法），其中 create table as 是指打造出 SAS 表格，如果省略该子句，结果就会直接输出到 SAS 的 output，select 则是要搜寻变量，可以想成是前面语法中的 keep，from 表示由哪些文档挑出变量，注意这边说明的是哪些文档，亦即我们可以在很多个文档里进行筛选的动作，where 则是条件句，其效果如同 if condition the output，group by 为分组进行写的语法，它是搭配在变量搜寻时使用的，在我们撰写描述统计的搜寻语法时使用，order by 则是将搜寻出来的数据进行排序，此处特别说明，这些子句一旦在语法中出现，不得变更其先后顺序，但部分是可以省略的，接下来开始介绍 SQL 的语法使用。

我们先介绍一个典型的搜寻数据的用法，在本例中，我们搜寻的变量是中文字，目的在于展示 proc sql 语法和常用的搜索引擎工作的相似性。事件数据的导入语句如表 6-12 所示。

表 6-12

```
proc import datafile="D:\The Application of SAS in Financial Research\CH06\data\sql\event"
  dbms=xlsx
  out=a
  replace;
quit;
proc sort nodupkey data=a;by event;
/* 针对事件类型进行分析，本文档包含不同公司的相同类型事件*/
run;
```

首先，在 “D:\The Application of SAS in Financial Research\CH06\data\sql” 中的 event 文档中，提供了 2010 年所有沪深两市的事件宣告日，共计 1688 个事件，我们先将原始文档处理成只包含单一事件的文档，以便后续的各种教学。

两种 SQL 语句的比较如表 6-13 所示。

表 6-13

<pre>/*可以比较 new1 以及 new2 的差异*/ proc sql; create table new as select y as year, m as month, d as day, y*10000+m*100+d as date, stkcd_nm as stkcd, event as event_type from a where event contains "现金股息" order by event; quit;</pre>	<pre>/*可以比较 new1 以及 new2 的差异*/ proc sql; create table new2 as select y, m, d, y*10000+m*100+d as date, stkcd_nm, event from a where event contains "现金股息" order by event; quit;</pre>
---	---

在程序撰写中，先声明本例子要搜寻的逻辑库所在位置，在该逻辑库中有一个名为 event 的文档，在 proc sql 的语法中，大部分的语法使用逗号来分隔，而不用分号来分隔，该程序语法总共有 4 大部分。第 1 部分：create table new as 是声明要创建一个新的表格；第 2 部分：as select y as y，直到 event as event (注意没有逗号)，是说挑选 y 变量作为新文档的 y 变量，……第 3 部分：from a 是指由 a 这个文档来搜寻；最后 where event contains ‘现金股息’是我们的搜寻条件，要 event 变量包含现金股息时就输出到 new1 以及 new2。

我们来看看 a 文档跟 new1 文档之间内容的区别是什么。

原始数据文件如图 6-4 所示，Table a 总共包含了 14 笔观测值，其中现金股利有 3 个数据，接着来看看 SQL 是否真的有搜索出有现金股息的数据，现金股息的搜索结果如图 6-5 所示。

	event	stkcd_nm	y	m	d
1	上海合并事件	600179 黑化股份	2010	12	31
2	上海回购注销	600315 上海家化	2010	5	21
3	上海增发	600000 浦发银行	2010	10	16
4	上海无偿配股	600000 浦发银行	2010	6	4
5	上海现金股息	600000 浦发银行	2010	6	4
6	上海配股	600036 招商银行	2010	3	2
7	下市合并事件	600263 路桥建设	2010	12	31
8	下市无偿配股	000527 美的电器	2010	5	7
9	下市现金股息	000522 白云山A	2010	7	22
10	深圳回购注销	000012 南玻A	2010	1	22
11	深圳增发	000001 平安银行	2010	7	3
12	深圳无偿配股	000006 深振业A	2010	5	7
13	深圳现金股息	000002 万科A	2010	5	10
14	深圳配股	000759 中百集团	2010	1	21

图 6-4

	y	m	d	date	stkcd_nm	event
1	2010	6	4	20100604	600000 浦发银行	上海现金股息
2	2010	7	22	20100722	000522 白云山A	下市现金股息
3	2010	5	10	20100510	000002 万科A	深圳现金股息

图 6-5

使用 SQL 语法以后，不但创造了我们想要的变量，同时也将 event 变量中有现金股利的字眼都输出来了，读者一定会有所疑问，难道不能在 data set 的过程中，挑选出我们需要的数据吗？我们以如表 6-14 所示的数据读取的程序语法来解释。数据读取法选取的结果如图 6-6 所示。

表 6-14

<pre>data newb; set a; if event="上海现金股息" then output; if event="下市现金股息" then output; if event="深圳现金股息" then output; run;</pre>
--

	event	stkcd_nm	y	m	d
1	上海现金股息	600000 浦发银行	2010	6	4
2	下市现金股息	000522 白云山A	2010	7	22
3	深圳现金股息	000002 万科A	2010	5	10

图 6-6

我们必须完全正确地写下各个变量里的文字才能输出所有的数据，但有时数据整理得并不是那么正确，如果变量内容为「上海现金股息」或者是「上海现金股息」，后者在辨识上不会等于「上海现金股息」，因为空白也是一种文本符号，我们要尝试很多种组合才能成功，因此还是建议读者采用该方法产生表格。

在挑出样本数据之后，另外要处理的就是，将这些样本的数据全部输出成一个文档，我们的目标只是要探讨该发现金股利的公司的股票报酬率数据，如果利用合并输出的方式，当母体数据非常大的时候，SAS 处理的速度就会相当缓慢，此时可以采用如表 6-15 所示的股票报酬率文档读取语句的方法来处理。

表 6-15

```
data price;
length stkcd $6.;
length nm $10.;
/*
先行声明 stkcd 是文本变量 长度为 6
nm 是文本变量 长度为 10
*/
infile "D:\The Application of SAS in Financial Research\CH06\data\sql\price.txt" dlm="09"x firstobs=2 missover;
/*
dlm 是告诉 SAS 数据的分隔符号是采用什么形式
'09'x 是表示数据间是使用 tab 键分隔的
*/
input stkcd $ nm $ date ret turnover mv;
/*股票代码 名称 日期 回报率 换手率 市值*/
y=int(date/10000);
m=mod(int(date/100),100);
drop nm;
/*在分析过程中 只需要股票代码，不需要股票名称*/
run;
data new2;
set new2;
stkcd=substr(stkcd_nm,1,6);
/*为了搭配股价文档的 stkcd 所以用文本处理出前六位*/
```

续表

```
run;  
  
proc sql;  
create table a as select *  
from price  
where stkcd in  
(select stkcd from new2);  
  
quit;
```

该语法的意义是当我们要挑选出 price 中所有的变量时，而要求的条件是在包含于事件样本中所有的股票代码的数据，假设我们的样本数据库有 10 万家公司，但是我们仅需要探讨其中 100 家公司，使用该语法可以相当快速地筛选出我们需要的数据，但是我们未必需要所有的变量，如果我们仅需要股票报酬率与市值的数据，就可以采用如表 6-16 所示筛选符合样本的股票代码语句的语法。

表 6-16

```
proc sql;  
create table b as select y, m, stkcd, ret, mv  
from price  
where stkcd in  
(select stkcd from new2);  
  
quit;
```

但是有时候我们需要其满足多个条件才能筛选出样本，我们并非只是要求相同的公司，也是需要是相同月份，此时可以用如表 6-17 所示的筛选样本以及其时间的报酬率数据的语法来撰写。

表 6-17

```
proc sql;  
create table c as select y, m, stkcd, ret, mv  
from price  
where stkcd in (select stkcd from new2) and  
y in (select y from new2) and  
m in (select m from new2);  
  
quit;
```

采用上述语法，会发现搜寻出来的数据是有问题的，因为 SAS 是将 stkcd 符合条件就输出，接着要求 y 符合条件就输出，最后输出符合条件的 m 数据。

反过来讨论，假设我们的目的是要删除发生发放现金股息的样本，则可以使用如表 6-18 所示的筛选非样本的报酬率文档的语法来执行。

表 6-18

```
proc sql;
create table nb as select  y , m,  stkcd, ret, mv
from price
where stkcd not in
(select stkcd from new2);
quit;
```

而 proc sql 亦可以进行条件式语法，在 data set 中，我们或许会撰写 if a<=0 then dummy=1;else dummy=0。在 SQL 的语法中，亦有类似的功能可以提供我们使用，虽然在撰写上不像 data set 语法中那样方便，但是在此处，笔者要特别强调，如果数据笔数较少，proc sql 的语法未必较利用 data set 或者 merge 处理好，因为利用 proc sql 的语法来撰写条件式语法是一件相当麻烦的事情，在 SQL 中使用条件语句的程序如表 6-19 所示。

表 6-19

```
proc sql;
create table d as select  y , m,  stkcd, ret, mv,/*此处加了逗号*/
case
when ret>5 then 1
when 5>=ret>0 then 2
when 0>=ret>-5 then 3
else 4
end as aa /*此处未加逗号 因其为最后一个变量*/
from price
where stkcd not in
(select stkcd from new2);
quit;
```

其中 case 到 end as aa 的语法就如同 data set 中的语法结构,如表 6-20 所示的数据步中的条件语句。

表 6-20

```
if ret>5 then aa=1;
else if 5>=ret>0 then aa=2;
else if 0>=ret>-5 then aa=3;
else aa=4;
```

如果我们要创造出文本变量，则可以利用如表 6-21 所示的 SQL 创建变量为文本语句（哑变量）的语法结构来撰写。

表 6-21

```
proc sql;
create table e as select  y , m,  stkcd, ret, mv,/*此处加了逗号*/
case
when ret>5 then 1
when 5>=ret>0 then 2
when 0>=ret>-5 then 3
else 4
end as aa, /*此处加了逗号*/
case
when y<2010 then "y2009"
else "y2010"
end as y2009 /*此处未加逗号 因它是最后一个变量*/
from price
where stkcd not in
(select stkcd from new2);
quit;
```

其撰写方式与使用 if then 的结构一样，我们可以利用该语法撰写在任意条件下，设定新变量的结果。

在上述的语法中，我们在撰写时都是以一个文档为条件，取得另外一个文档的变量以及数据的，实际上，SQL 亦提供了同时搜寻多个文档的方法，并且产生出符合条件的文档数据，如表 6-22 所示。

表 6-22

```
proc sql;
create table f
as select
a.y as y,
a.m as m,
a.ret as ret,
a.stkcd as stkcd,
b.event as event
from price a ,new2 b
/*revise 20150628*/
where a.stkcd=b.stkcd and
/*revise 20150628*/
a.y=b.y and
a.m=b.m;
quit;
```

该程序会从 price 搜寻出 y、m、stkcd、ret 等变量数据，并且由 new 搜寻出 event 的数据，此处虽然只是以两个文档为例，实质上仍可使用两个以上的文档进行搜索，只是条件将会变得相当复杂，基本上仍然建议对两个文档进行搜索为好，然而，如果我们并没有打算更改变量，上述语法也可以改写，如表 6-23 所示。

表 6-23

```
proc sql;
create table g
as select  a.y, a.m, a.ret, a.stkcd, b.event
from price a, new2 b
where a.stkcd=b.stkcd and
a.y=b.y and
a.m=b.m;
quit;
```

注意：撰写 SQL 语法时，请特别注意逗号，本例中最后一个变量以及最后一个文档标号都没有加逗号，并非是笔者的疏忽，而是语法上撰写的要求。

接下来，我们介绍一直没有提到的 order by 以及 group by 子句，这两个子句的功能在 SAS 中可以替代许多功能，在熟悉这两个子句功能后，可以节省许多程序运算的时间。

创建虚拟数据如表 6-24 所示。其虚拟数据值如图 6-7 所示。

表 6-24

```
data a;
do i= 1 to 100;
do j=1 to 10;
a=rannor(1);
output;
end;
end;
run;
```

	i	j	a
1	1	1	1.8048229506
2	1	2	-0.079915021
3	1	3	0.396576855
4	1	4	-1.083317655
5	1	5	2.2382943651
6	1	6	-0.624232294
7	1	7	0.5136577083
8	1	8	-0.086609117
9	1	9	-0.594178733
10	1	10	0.0318908181
11	2	1	-0.737798572
12	2	2	-0.250139175
13	2	3	0.6850047647
14	2	4	-0.804158132
15	2	5	-0.74428108
16	2	6	-0.795502822
17	2	7	0.3407105493
18	2	8	-0.300509804
19	2	9	-1.349846516
20	2	10	0.432704861
21	3	1	1.3057162426
22	3	2	1.4251270104
23	3	3	-0.415801019

图 6-7

执行完该程序以后，可以得到 1000 笔数据，如果我们想计算每个相同 j 底下，a 的平均值，要如何进行呢？虽然在后面章节的语法中，我们可以采用 proc means 的程序来计算，但在此处我们利用到目前为止章节介绍的语法（如表 6-25 所示的模拟数据的均值语句）来撰写这样的程序。

表 6-25

```
data a;  
do i= 1 to 100;  
do j=1 to 10;  
a=rannor(1);  
output;  
end;  
end;  
run;  
proc sort data=a;by j;  
run;  
proc transpose data=a out=b;  
var a;  
by j;  
run;  
data b;  
set b;  
a=mean(of coll-col100);  
keep j a;  
run;
```

用这样的语法，我们可以得到每个 j 的平均值（如图 6-8 所示的计算 J 组的均值结果），但是有一个缺点，那就是我们必须要知道有几个观测值，在本例中，我们因为知道 i=1 to 100，故可以确定数据转置后，会有 100 个变量，否则根本无从进行，当然这样的困扰在描述统计的章节中是不存在的。

	j	a
1	1	-0.196211779
2	2	0.1366000866
3	3	-0.07524906
4	4	0.0262499839
5	5	-0.169685207
6	6	-0.04641275
7	7	0.0804283583
8	8	0.0312312292
9	9	0.0667195359
10	10	0.1812595547

图 6-8

接下来，学习如何使用 SQL 的语法来进行这样的工作。在前面我们采用了 $a.y*10000+a.m*100+a.d$ as date 的语法，与 data set 中的 $date=y*10000+m*100+d$ 的意思一样，mean、sum 这些语法，也可以灵活运用 SQL 中。以 SQL 计算 J 组均值的语句如表 6-26 所示，SQL 均值运算的结果如图 6-9 所示。

表 6-26

```
proc sql;
create table c
as select i,j, a,
mean(a) as meana
from a
group by j;
quit;
```

94	94	1	-0.503345357	-0.196211779
95	95	1	2.3686389722	-0.196211779
96	96	1	1.1687903827	-0.196211779
97	97	1	-2.896054038	-0.196211779
98	98	1	0.4409047944	-0.196211779
99	99	1	0.7858028796	-0.196211779
100	100	1	0.7001667366	-0.196211779
101	1	2	-0.079915021	0.1366000866
102	2	2	-0.250139175	0.1366000866
103	3	2	1.4251270104	0.1366000866
104	4	2	-0.635758247	0.1366000866
105	5	2	-2.331337587	0.1366000866
106	6	2	-0.055347321	0.1366000866
107	7	2	1.6059491196	0.1366000866

图 6-9

就 SQL 均值运算的结果来看，我们确实能得到均值，但是却得不到 10 个均值的平均数值，其实这样的做法相当好用，在财务研究中，常会需要计算扣除平均值调整后的数值，如果要计算调整后的数值，则可以用如表 6-27 所示的以 SQL 计算均值调整后语句的语法。均值调整后的运算结果如图 6-10 所示。

表 6-27

```
proc sql;
create table d
as select i,j, a,
a-mean(a) as adj
from a
group by j;
quit;
```

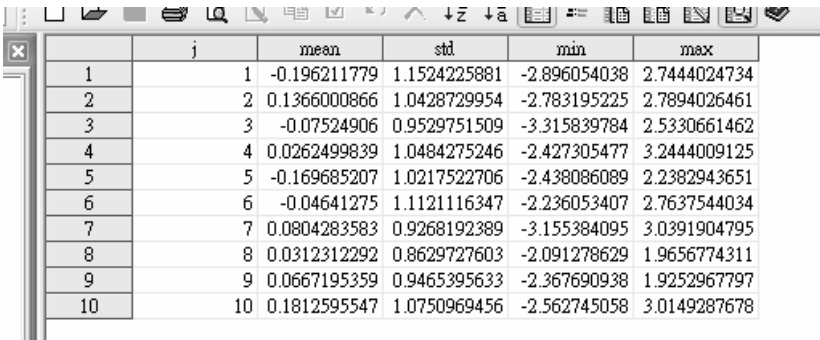
	i	j	a	adj
1	1	1	1.8048229506	2.0010347296
2	2	1	-0.737798572	-0.541586793
3	3	1	1.3057162426	1.5019280216
4	4	1	-0.630841419	-0.43462964
5	5	1	-1.359501933	-1.163290154
6	6	1	0.3489217786	0.5451335575
7	7	1	-0.346431605	-0.150219826
8	8	1	0.3474949211	0.5437067
9	9	1	1.3786037626	1.5748155415

图 6-10

如此一来，我们就轻松算完调整后的数据了，但是如果只是要简单地计算平均数据，而不需要得到每一笔数据，要如何进行呢？我们以均值、标准偏差、最大值、最小值为例，用 SQL 求出描述性统计的语句如表 6-28 所示。描述性的统计计算结果如图 6-11 所示。

表 6-28

```
proc sql;
create table e
as select j,
mean(a) as mean,
std(a) as std,
min(a) as min,
max(a) as max
from a
group by j;
quit;
```



	j	mean	std	min	max
1	1	-0.196211779	1.1524225881	-2.896054038	2.7444024734
2	2	0.1366000866	1.0428729954	-2.783195225	2.7894026461
3	3	-0.07524906	0.9529751509	-3.315839784	2.5330661462
4	4	0.0262499839	1.0484275246	-2.427305477	3.2444009125
5	5	-0.169685207	1.0217522706	-2.438086089	2.2382943651
6	6	-0.04641275	1.1121116347	-2.236053407	2.7637544034
7	7	0.0804283583	0.9268192389	-3.155384095	3.0391904795
8	8	0.0312312292	0.8629727603	-2.091278629	1.9656774311
9	9	0.0667195359	0.9465395633	-2.367690938	1.9252967797
10	10	0.1812595547	1.0750969456	-2.562745058	3.0149287678

图 6-11

这样就取得均值、标准偏差、最小值与最大值的数据了，语法上的最主要差异在于我们只撰写跟 j 有关的变量。要注意 SQL 的语法逻辑在于查询，当我们查询的指令下得越多，输出的结果就会将符合条件的数据都输出，如果我们只查询 j 与均值有关的这些数据，便会输出最精简的结果，但是如果

多查询了 i 以及 a 的变量，则会输出 1000 笔观测值，而其中相同的 j 都会有相同的均值等变量。此处要特别说明，在 SQL 采用分组计算数值中，median 以及 geomean 是无法作用的，但是由于几何均值在财务研究中要计算复利报酬率相当重要，所以我们仍然可以结合其他方法来计算 geomean 的数据，但 median 以及 geomean 仍然是可以使用的语法，只是其语法撰写形式要用如表 6-29 所示的 SQL 中计算算术平均、几何平均与中位数语句的程序。算术平均、几何平均与中位数运算的结果如图 6-12 所示。

表 6-29

```
proc sql;
create table f
as select  i,j,a,
mean(i,j,a) as mean,
geomean(i,j,a) as geo,
median(i,j,a) as median
from a;
quit;
```

	i	j	a	mean	geo	median
1	1	1	1.8048229506	1.2682743169	1.2175258806	1
2	2	1	-0.737798572	0.7540671426	.	1
3	3	1	1.3057162426	1.7685720809	1.5763646495	1.3057162426
4	4	1	-0.630841419	1.4563861937	.	1
5	5	1	-1.359501933	1.546832689	.	1
6	6	1	0.3489217786	2.4496405929	1.2792628144	1
7	7	1	-0.346431605	2.551189465	.	1
8	8	1	0.3474949211	3.1158316404	1.4060890175	1
9	9	1	1.3786037626	3.7928679209	2.3150514981	1.3786037626
10	10	1	-0.052306207	3.6492312644	.	1
11	11	1	0.2525898478	4.0841966159	1.4058409584	1
12	12	1	0.9942629953	4.6647543318	2.2850419319	1
13	13	1	0.7711793849	4.9237264616	2.156252361	1
14	14	1	0.3271102761	5.1090367587	1.6606345697	1
15	15	1	0.2501110000	5.0760617000	.	1

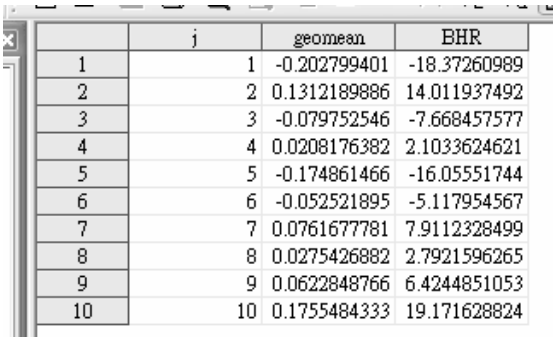
图 6-12

由图 6-12 可见，在 SQL 中 median 以及 geomean 这两个语法仅能做变量间的处理，而无法进行变量的分组处理。不过，在描述统计的语法中，我们可以轻易地算出中位数，唯有分组的几何均值，如果不能使用 SQL 语法处理，使用其他的方法来进行都相当复杂，我们在 SQL 中可以利用做分组计算的三个函数来处理，log、exp 以及 mean，如果不计算平均，则可采用 log、exp 以及 sum 来处理。

我们假设变量 a 是报酬率，分别计算平均报酬率以及买进持有报酬率，在美国的数据中，往往会取得日报酬率，而没有月报酬率，在文献中将会出现的是将日报酬率复利计算成月报酬率，也就是计算买进持有报酬率。以 SQL 实现复利报酬率的语句如表 6-30 所示。复利报酬率运行的结果如图 6-13 所示。

表 6-30

```
proc sql;
create table f
as select j,
(exp (mean (log(1+0.01*a)))-1)*100 as geomean,
(exp(sum(log(1+0.01*a)))-1)*100 as BHR
from a
group by j;
quit;
```



	j	geomean	BHR
1	1	-0.202799401	-18.37260989
2	2	0.1312189886	14.011937492
3	3	-0.079752546	-7.668457577
4	4	0.0208176382	2.1033624621
5	5	-0.174861466	-16.05551744
6	6	-0.052521895	-5.117954567
7	7	0.0761677781	7.9112328499
8	8	0.0275426882	2.7921596265
9	9	0.0622848766	6.4244851053
10	10	0.1755484333	19.171628824

图 6-13

最后，我们介绍 order by 这个子句。严格来说，order by 的做法与 proc sort 一模一样，但是如果我们在处理数据的过程中，已经确定要将数据排序，与其先用 data set 处理数据，再使用 proc sort 将数据排序，不如直接利用 proc sql 搜寻必要的数 据，并且顺便将数据排序，前者是两个步骤，但是这两个步骤会重复读取一笔数据；而使用 proc sql，则只会花费读取一次数据的时间。下面简单地将数据进行排序，如表 6-31 所示。

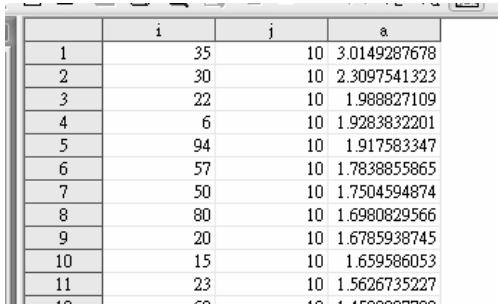
表 6-31

```
proc sql;
create table g
as select *
from a
order by j,a;
quit;
```

当然在 SQL 语法中，也可以将数据由大到小排序，只是语法是用 descending 的前 4 个字母，而且顺序和 proc sort 相反，在利用 proc sql 语法时，要特别注意该部分的顺序，避免程序撰写错误，如表 6-32 所示。降幂排序的结果如图 6-14 所示。

表 6-32

```
proc sql;
create table h
as select *
from a
order by j desc,a desc;
quit;
```



	i	j	a
1	35	10	3.0149287678
2	30	10	2.3097541323
3	22	10	1.988827109
4	6	10	1.9283832201
5	94	10	1.917583347
6	57	10	1.7838855865
7	50	10	1.7504594874
8	80	10	1.6980829566
9	20	10	1.6785938745
10	15	10	1.659586053
11	23	10	1.5626735227

图 6-14

至此，我们最后再仔细思考这样的问题：如果今天我们要水平合并两个文档，要根据这两个文档的公司代码、年分、月份进行数据合并的动作。在前面的章节中，我们会利用 proc sort 的方法，将两个文档各自依照相同的变量排序，最后再合并两个文档，就数据处理而言，这是绝对正确的模式，但是在速度上比 proc sql 要慢，因为在 SQL 语言中，我们只要利用条件句 where a.var1=b.var1 and a.var2=b.var2 and a.var3=b.var3 就可以顺利合并了，而且只需要读取一次文档，如果需要常常整理数据，可以好好地学一下 SQL，以方便进行研究。在 SQL 中进行计算几何平均报酬率与买进持有报酬率的语法如表 6-33 所示。

表 6-33

语 法	含 义
(exp (mean (log(1+0.01*a)))-1)*100	计算几何平均报酬率
(exp (sum (log(1+0.01*a)))-1)*100	计算买进持有报酬率

6.3 总结

本章介绍了如何在 SAS 中管理逻辑库以及搜寻需要的数据，如果熟悉这两种程序，对于进行数据管理会相当有帮助，如果不熟悉，本质上亦不会影响使用 SAS 的效能，在整理数据上亦可以借助于撰写程序达到相同的结果，但如果深入了解，则可节省相当多的时间。

第 7 章

宏语法（%macro）

到目前为止，本书介绍了整理数据所需要的方法，大部分的数据处理都不会脱离前面的章节所介绍的方法，只要读者在取得数据的时候，好好观察和了解你的数据，再来思考如何整理数据，并且累积足够的经验，便能根据研究目的准确地选择处理该数据所需要的语法。

接下来我们将介绍一些实证上会使用到的语法，例如：描述统计、相关语法、回归语法，等等。在 SAS 中使用这些语法虽然简便，但是如果读者需要进行更多的研究以及建立数学模型，就必须要为这些数据以及模型重新撰写程序语法，从而产生了些许麻烦，此时使用宏语法，就可以很方便地重复执行相同的程序，而熟练的 SAS 用户，更可以利用宏语法写出一套标准化程序，只要数据格式正确，就可以快速地完成实证任务。

7.1 基础宏语法

在撰写 SAS 语法时，宏是相当烦杂的写法，许多初学者都认为宏是相当困难的，事实上，只要能够掌握一些基础观念，以下以 example_7_1.sas 为例来介绍基础的宏语法。取一虚拟的乱数，如表 7-1 所示。

表 7-1

<pre>data a; do i=1 to 100; a=rannor(1); b=ranuni(1); output; end; run;</pre>

首先，我们产生一笔数据，该数据包含 a 与 b 两个变量，假设我们已经知道有另外一个文档 a，其也有 a、b 两个变量而且都要进行相同的整理，由于数据处理的流程一致，我们并不需要为另外一个文档撰写语法，而是撰写一个可以重复使用的宏语法来进行快速简便的处理，基础的宏语法如表 7-2 所示。

表 7-2

<pre>%macro a; data a; set a; c=a+b; run; %mend; %a;</pre>
--

在该部分，我们撰写一个简单的宏 a，%macro a 为声明撰写一个宏语法，该数据及名称为 a，接下来的部分为该宏要执行的程序，是要求 SAS 读取 a 并计算一个变量 c，其等于 a+b，最后%mend 为告知 SAS 该宏语法撰写完毕。然而，在我们将该部分语法送出执行时，SAS 并不会执行宏语法里面的程序，这部分的语法仅仅是告知 SAS 我们撰写了一个宏，临时先储存起来，接下来我们要执行这部分的语法时，可以呼叫出宏来执行程序。

当我们要执行宏语法时，仅需要输入%a，这里的 a 指的是之前所撰写的%macro a 里面的 a，亦即呼叫宏 a，如果读者自行撰写了其他的宏，可自行命名，宏语法读取程序执行该语法的 log 文件如图 7-1 所示。

```
15 %macro a;  
16 data a;  
17 set a;  
18 c=a+b;  
19 run;  
20 %mend;  
  
21 %a;  
  
NOTE: There were 100 observations read from the data set WORK.A.
```

图 7-1

如 log 文件所示，在%mend 与 a 之间有一个空格，这显示 SAS 是将这两段指令分开处理，而只有在 a；之后才有执行语法的动作。接下来我们撰写另外的 a 文档来执行宏 a 的语法，如表 7-3 所示。

表 7-3

<pre>data a; a=2; b=3; run; %a;</pre>

在该部分，我们没有撰写%macro 到%mend 的语法，而是在产生数据 a 以后便直接输入%a，接下来我们来看程序执行日志文件的结果，如图 7-2 所示。

```
22 data a;  
23 a=2;  
24 b=3;  
25 run;  
  
NOTE: The data set WORK.A has 1 observations and 2 variables.  
NOTE: DATA statement used (Total process time):  
      real time           0.04 seconds  
      cpu time            0.01 seconds  
  
26 %a;  
  
NOTE: There were 1 observations read from the data set WORK.A.  
NOTE: The data set WORK.A has 1 observations and 3 variables.  
NOTE: DATA statement used (Total process time):  
      real time           0.03 seconds  
      cpu time            0.00 seconds
```

图 7-2

由日志文件来看，SAS 的确是使用%a 来取代那一小串的程序语法，这里提供了一个小小有趣的地方，如果读者能够将程序撰写得相当的标准文，就可以采用宏语法来执行程序，而且可以大幅度地节省程序编辑空间。然而，在本例中严格要求文档名必须为 a，仍然有些不便，因此我们将该宏程序进行一些修正，以便让该宏程序更好地应用，创建宏变量的宏语法介绍如表 7-4 所示。

表 7-4

<pre>%macro b(in,out); data &out; set &in; c=a+b; run; %mend; data dd; a=100; do i=1 to 100; b=rannor(1); output; end; run;</pre>

该部分将原有的程序语法做了局部的修正，在此处我们另外撰写了宏 b，但是在其后加入了（in, out），此处的做法是在宏的语法里面，我们做了一个指定变量的处理，在程序中若有&in 或者&out，就会直接取代为我们所指定的 in 以及 out,在此处 in 指的是读取的文档名,out 指的是最后存取的文档名，另外，我们产生一个 dd 文档，如果我们要执行宏 b 则可以输入%b(dd,b);。

此处的执行是告知 SAS 我们要读取的文档为 dd，将之存取为 b，至于先后顺序则是根据撰写者在宏里的程序语法决定的，当然我们也不一定只能指定两个变量 in 以及 out，根据宏的复杂程度，亦可用到 10 个以上的变量，而在某些情况下，变量名也仅仅是用来代表数字，所以在使用上要特别小心。宏变量执行的结果如图 7-3 所示。

```
38 %b(dd,b);  
NOTE: There were 100 observations read from the data set WORK.DD.  
NOTE: The data set WORK.B has 100 observations and 4 variables.  
NOTE: DATA statement used (Total process time):  
      real time           0.03 seconds  
      cpu time            0.00 seconds
```

图 7-3

宏语法的简单介绍如表 7-5 所示。

表 7-5

语 法	含 义
%macro mname;	声明要撰写一个宏语法，mname 为宏名
%mend;	宏语法结束
%mname;	要求 SAS 执行命名为 mname 的宏
%macro mname(var1,var2,...);	在宏语法中要指定变量名以进行标准化程序语法
%mname(var1,var2,...);	执行使用指定变量的宏

7.2 进阶宏语法

在 7.1 节中，我们介绍了基础的宏观念，亦即撰写一个标准化的程序语法以便让不同文档执行相同程序，在本节中，我们采用 example_7_2.sas 的例子来介绍这部分的语法。附加多文档的宏语法如表 7-6 所示。

表 7-6

libname aa 'D:\The Application of SAS in Financial Research\SAS data\daily data';			
%macro a(a);			
%do i=2002 %to 2006;			
proc append base=&a data=aa.d&i;			
quit;			
%end;			
%mend;			
%a(price);			

这个宏是专门用来处理存取数据文件中具有连续性代码的状况，笔者在存放文档时，通常是采用年度为依归，即将数据一年一年地存放，这样就可以顺利地重复使用该宏语法。文档命名范例如图 7-4 所示。

名称	修改日期	类型	大小
d2001	2015/6/15 21:14	SAS Data Set	22,144 KB
d2002	2015/6/15 0:28	SAS Data Set	23,808 KB
d2003	2015/6/15 0:28	SAS Data Set	24,640 KB
d2004	2015/6/15 0:28	SAS Data Set	24,832 KB
d2005	2015/6/15 0:28	SAS Data Set	24,384 KB
d2006	2015/6/15 0:28	SAS Data Set	24,256 KB

图 7-4

在如表 7-6 所示的宏语法中，我们新增了%do i=2002 %to 2006 以及%end 这两个语句，这在 SAS 中亦是一个配对的指令，有%do 就要有%end 来作为结束，如同在前面的部分有 do 以及 end 亦是相呼应的语法。在该宏中，SAS 会由 2002 开始进行到 2006，由于指定了 i=2002 %to 2006，在宏里面一样使用&i 来命名这些变量，在整个宏中，SAS 一开始会先执行 2002，先将 aa.d2002 的数据复制成&a 数据，在本例中&a 以 price 取代之，接着再将 aa.d2003、aa.d2004、aa.d2005、aa.d2006 依序合并到 price 的数据后面。

然而，如果我们在后续的程序出错，必须要重新合并 price 的文档时，由于附加语法的缘故，必须先删除掉文档才能进行合并的工作，因此该语法不算是完整的宏语法。在第 6 章中，我们提到了 proc datasets 这个语法，该语法可以删除掉文档，故我们可以将 proc datasets 的 delete 纳入宏中，如表 7-7

所示的另一个附加多文档的宏语法。

表 7-7

```
%macro b(b);
proc datasets noprint;
delete &b;
quit;
%do i=2002 %to 2006;
    proc append base=&b data=aa.d&i;
    quit;
%end;
%mend;
%b(price);
```

我们将删除语法新增到%do i=2002 %to 2006 之前，在合并语法之前就将之删除，该语法绝对不能写在%do i=2002 %to 2006 之后，这会变成在&i=2002 时就会删除文档，&i=2003 亦会删除文档，依序都会先删除再合并，但如果写在%do i=2002 %to 2006 之前，则会先行删除文档后，然后开始执行&i 从 2002 到 2006 的语法。

该语法亦可做更进一步的标准化过程，对于读者而言，不一定会使用 2002 到 2006 的数据，可能想使用 2001 到 2005 或者由于其他条件而需要的不同年份数据，如果使用宏 b，在使用时就必须要更改原有的宏，因此我们可以另外设定两个指定变量，来设定开始和终止的年份，附加多文档宏语法的设计如表 7-8 所示。

表 7-8

```
%macro c(ys, ye, c);
proc datasets noprint;
delete &c;
quit;
%do i=&ys %to &ye;
    proc append base=&c data=aa.d&i;
    quit;
%end;
%mend;
%c(2001, 2003, price);
```

至此，我们就可以根据我们所需要的年份去建立自己需要的文档，如果数据完全是标准化的格式，只需要修改 libname（数据库名）的路径，就可以快速地用同一个宏语法合并需要的数据。

在前面的部分我们使用%do i=2002 %to 2006 来进行不同年份的指定工作，但是如果在同一个逻辑库中，其文档并非以年代区分，而可能是不同的股票，或者存取方式没有采取连续性格式的文档，那

么我们该如何处理呢？以下采用一个如表 7-9 所示的虚拟多个无命名规律文档的简单的例子来加以探讨。

表 7-9

<pre>data a b c d e f i j k; do i=1 to 100; a=rannor(20); output; end; run;</pre>

我们先产生 a、b、c、d、e、f、i、j、k 9 个文档，如果我们要合并这些文档，可以采用两种方法。垂直合并多文档的语法如表 7-10 所示。

表 7-10

<pre>data final; set a b c d e f i j k; run;</pre>
--

该方法当然可以顺利地合并文档，因为只有 9 个文档，然而在实际上我们接触的文档并不一定会像这个例子一样只有简单的文档名并且只有 9 个，而如果有上千个文档，仍然会造成不便，为此我们必须要知道文档内容是什么。这就是 Contents 程序步的功能，其程序如表 7-11 所示。

表 7-11

<pre>proc datasets noprint; delete final; quit; proc contents data=work._all_ noprint out=file; /* 数据库._all_ 表示要检查该逻辑库下所有的文档内容 work._all_ all the file in work library aa._all_ all the file in aa library */ quit;</pre>

也就是说，一样使用 proc datasets 这个语法，先将 final 这个文档删除掉，接着使用 contents（内容）这个数据库的程序步，在 7.1 节我们并未提到该语法，基本上当执行该语法时，我们可以检查数据里面的各项内容，contents 程序步的输出结果如图 7-5 所示。

逻辑库名	逻辑库成员名	数据集标签	特殊的数据集类型 (FROM TYPE=)	变量名称	变量类型	变量长度
WORK	A			a	1	8
WORK	A			i	1	8
WORK	B			a	1	8
WORK	B			i	1	8
WORK	C			a	1	8
WORK	C			i	1	8
WORK	D			a	1	8
WORK	D			i	1	8
WORK	E			a	1	8
WORK	E			i	1	8
WORK	F			a	1	8
WORK	F			i	1	8
WORK	I			a	1	8
WORK	I			i	1	8
WORK	J			a	1	8
WORK	J			i	1	8
WORK	K			a	1	8
WORK	K			i	1	8

图 7-5

检验 file 的文档可以发现，我们自动取得了 work 这个逻辑库里所有的 tables（A~K）文档及其所有的变量数据，但是我们只需要合并一次文档，因此要利用 proc sort 的删除重复值的功能来处理，如表 7-12 所示。待合并的逻辑库成员名如图 7-6 所示。

表 7-12

```
proc sort data=file(keep=memname) nodupkey;by memname;
run;
```

在准备好数据之后，我们便利用该文档中的 filename（文件名）变量，作为我们呼叫与其变量同名的 SAS 文档，进行附加文档的语句如表 7-13 所示。

逻辑库成员名	
1	A
2	B
3	C
4	D
5	E
6	F
7	I
8	J
9	K

图 7-6

表 7-13

```
%macro d(final,n);
proc datasets noprint;
delete &final;
quit;
%do i=1 %to &n;
data _null_;
```

```
set file;
  if _n_=&i then do;
    call symput('ss', memname);
  end;

run;
proc append base=&final data=&ss;
quit;
%end;
%mend;
%d(final,9);
```

在本例中，我们使用了一个虚拟数据文件_null_，在 SAS 中该文档名是用来临时处理数据用的，一旦程序执行完毕，该虚拟数据文件就会被直接删除，接着我们进行一个条件语法，满足其条件时就执行 call symput('ss', memname)，在此处表示设定一个&ss 变量会等于 memname 里的变量，因此当&i=1 时，&ss 自动执行为 a，&i=2 时，&ss 会自动执行为 b，但是一旦&i 大于该逻辑库的总数，超过的部分都会自动默认为最后一笔数据，在本例中如果&i>9，&ss 皆为 k，因此当我们利用程序进行合并时，SAS 会自动抓取 a、b、c、d、e、f、i、j、k 9 个文档来进行合并的。

接下来我们把原本的宏稍微修改一下，以方便读者进一步了解&SS 的使用方法。采用双引号的宏与采用单引号的宏的区别如表 7-14 所示。

表 7-14

<pre>%macro d(final,n); proc datasets noprint; delete &final; quit; %do i=1 %to &n; data _null_; set file; if _n_=&i then do; call symput('ss', memname); end; run; proc append base=&final data=&ss; quit; %end; %mend; %d(final,9);</pre>	<pre>%macro dl(final,n); proc datasets noprint; delete &final; quit; %do i=1 %to &n; data _null_; set file; if _n_=&i then do; call symput('ss', memname); end; run; data &ss; set &ss; file1="&ss"; file2='&ss'; run;</pre>
---	--

续表

	/*注意 单引号 以及双引号的差异*/ proc append base=&final data=&ss; quit; %end; %mend; %d1(final2,9);
--	--

在 d1 的宏中，加入了 file1=” &ss” 以及 file2=’&ss’，其中 file1 会以 A~K 呈现，file2 会以&ss 呈现，因此在使用上要用 “&ss” 才能够将其读进栏位中。单引号与双引号语法的差异如图 7-7 所示。

	i	a	file1	file2
1	1	-0.689820438	A	&ss
2	2	0.0318787669	A	&ss
3	3	-0.974736784	A	&ss
4	4	-0.366802589	A	&ss
5	5	-0.044437637	A	&ss

图 7-7

该部分的宏语法看似简单，但是其运用层面相当的广，首先 proc contents 可以读取特定逻辑库中所有的文档以及其所有的变量栏位，当读者进行美国金融数据研究时，会由 CRSP 下载其股票数据，由 COMPUSTAT 下载其财务报表数据，但其文档命名缺乏顺序性，例如图 7-8 所示的 compustat 数据集图。

CRSP Daily Stock	2011/1/3 11:13	Adobe Acrobat ...	67 KB
CRSP_D_00_02	2011/1/22 8:05	SAS Data Set	2,611,808...
CRSP_D_03_05	2011/1/22 8:35	SAS Data Set	2,288,088...
CRSP_D_06_08	2011/1/22 9:13	SAS Data Set	2,300,288...
CRSP_D_09_09	2011/1/22 9:44	SAS Data Set	733,728 KB
CRSP_D_10_12	2013/12/16 15:54	SAS Data Set	2,515,761...
CRSP_D_25_50	2011/1/22 2:38	SAS Data Set	2,524,928...
CRSP_D_51_60	2011/1/22 2:56	SAS Data Set	1,200,488...
CRSP_D_61_70	2011/1/22 3:18	SAS Data Set	2,234,248...
CRSP_D_71_75	2011/1/22 4:30	SAS Data Set	2,403,728...
CRSP_D_76_80	2011/1/22 4:47	SAS Data Set	2,845,888...
CRSP_D_81_85	2011/1/22 3:49	SAS Data Set	3,295,848...
CRSP_D_86_90	2011/1/22 4:12	SAS Data Set	3,860,608...
CRSP_D_94_96	2011/1/22 5:21	SAS Data Set	2,816,368...
CRSP_D_97_99	2011/1/22 7:15	SAS Data Set	2,978,968...
q3b79bdb48ee01f48	2011/1/22 5:08	SAS Data Set	2,358,248...
q790d8743be916ccf	2015/2/1 3:35	SAS Data Set	834,968 KB

图 7-8

其中以 CRSP 开头的文档名是笔者自行重新命名的，而 q3b79bd48ee01f48 以及 q790d8743be916ccf 则是 CSRP 下载后自动生成的文档名称，这些文档虽然可以自己一个一个地手动输入程序，但是借助

proc contents 的功能，绝对可以节省下编程的时间。

接下来我们介绍将前面几个章节使用的程序撰写成宏，可以大幅缩减撰写程序的时间。

首先，我们先撰写一个排序宏，就基本的语法而言，我们仅需要决定依照何种变量排序以及数据源为哪个文档，因此就可以产生出这个简单的排序宏，如表 7-15 所示。接下来我们可以撰写 proc transpose 以及 proc rank 的小宏。

表 7-15

<pre>%macro sort(sort,file); proc sort data=&file;by &sort; run; %mend; %sort(y m d,price);</pre>

转置宏语法如表 7-16 所示，在此处撰写了 proc transpose 的宏，可以将数据转置处理成一个单纯存放特定变量的公司月份的文档，观察其数据可以帮助我们快速地检查样本的特征值在序列上的数据特性。然而，其缺点则是所有的变量命名都变为 COL1-COLn，所幸我们仍可使用完整的月份数据来编码，之后进行一对一地配对时，就可以了解 COLn 是属于第几个月份的数据了。

表 7-16

<pre>%macro transpose(sort,file,var); proc transpose data=&file out=&var; var &var; by &sort; run; %mend; %sort(code,price); %transpose(code,price,ret); %transpose(code,price,mv); %transpose(code,price,turnover);</pre>
--

在该部分，我们介绍分组的小宏，如表 7-17 所示，细心的读者会发现在要求 SAS 执行宏时，我们的第一个值并未做任何指定，相应地在宏程序中，这是要求依照 sort 的变量来进行分组，基本上忽略该变量可能会让 SAS 出现警告（或者 SAS 自动修正错误，认定 by 是多余的），但实际上不会影响结果，用这个方法进行分组，SAS 是针对所有的样本分组，只是该宏的缺点是一次仅能针对一个变量进行分组，如果要做 dependent 或者 independent 的分组，就要多写几次，使用宏语法进行独立与相应分组和程序如表 7-18 所示。

表 7-17

<pre>%macro ranks(sort,file,var,groups); proc rank data=&file out=&file groups=&groups; var &var; ranks rank_&var; by &sort; run; %mend; %ranks(, price, turnover,5);</pre>
--

表 7-18

<pre>/*进行 dependent sort 的方法 */ %sort(rank_turnover,price); %ranks(rank_turnover , price, mv,5); /*进行 independent sort 的方法*/ %ranks(, price, turnover,5); %ranks(, price, mv,5);</pre>
--

到目前为止，我们介绍了一些撰写宏程序的方法，事实上撰写一个标准化的宏的好处在于可以重复利用，并且可以大幅度地精简 SAS 语法以及减少一些语法出错，甚至由于已经进行了标准化的宏程序，在结果出错时，我们通常也不太需要针对该部分语法进行除错。

在本节最后，我们介绍一个极简便的方法，首先，我们可以将本节中的宏 c、sort、transpose、ranks 一起存储到外挂宏这个文档。接下来皆可以用如表 7-19 所示的读取外部宏语法的方式来读取使用这些宏。

表 7-19

<pre>%include "D:\The Application of SAS in Financial Research\CH07\program_cn\special macro.sas"; libname aa 'D:\SAS 在财务研究上的运用\SAS data\日数据\除权息日报酬\上市公司'; %c(2001,2003,price); %sort(y m d,price); %ranks(, price, turnover,5); /*进行 dependent sort 的方法 */ %sort(rank_turnover,price); %ranks(rank_turnover , price, mv,5); /*进行 independent sort 的方法*/ %ranks(, price, turnover,5); %ranks(, price, mv,5);</pre>

在此处，使用%include (或者%inc)，然后依照读外部文本文档的方式，写入适当的路径，就可以让 SAS 直接去读取该文档了。

```
%include "D:\The Application of SAS in Financial Research\CH07\program\special macro.sas";
```

由于都是宏语法，所以 SAS 不会在一开始就进行运算，仅仅是将该程序读取进去而已，接下来我们就可以开始使用宏语法了。

以这样的方法，就可以完整地将本章一些重要的语法直接进行，而且在程序编辑上也比较简洁。

在前面的部分，我们介绍了基础的宏语法以及如何将之形成插件使用，在接下来的章节里有标准化的宏语法可以提供给读者使用，然后，我们将引入一些数据处理的方法。在财务研究中，最常遇到的情况就是要取得个股的前几个月的信息，这语法既简单又麻烦，我们以抓取前三个月的股票报酬率为例加以说明。利用宏语法抓取完整的报酬率文档的语法如表 7-20 所示。

表 7-20

<pre>libname aa "D:\The Application of SAS in Financial Research\SAS data\monthly price"; data a; set aa.price; run; %include "D:\The Application of SAS in Financial Research\CH07\program_cn\special macro.sas"; %sort(code y m,a);</pre>	
---	--

该语法相当简单，但是在此处仅指过去 3 个月的数据，如果我们需要用到过去 60 个月的数据，那所要输入的程序语法就会相当的冗长，一旦语法出错，在进行程序除错时就会相当麻烦，在此处我们撰写一个 SAS 可以使用的宏，宏语法抓取过去 N 个月报酬率的比较，如表 7-21 所示。

表 7-21

<pre>data a; set a; r1=lag1(ret); r2=lag2(ret); r3=lag3(ret); if code^=lag1(code) then r1=.; if code^=lag2(code) then r2=.; if code^=lag3(code) then r3=.; run;</pre>	<pre>%macro lag(file,code,var1,var2,n); %do i=1 %to &n; data &file; set &file; &var2&i=lag&i(&var1); if &code^=lag&i(&code) then &var2&i=.; run; %end; %mend; %lag(a,code,ret,r,20);</pre>
---	--

这样我们便简单地撰写出一个宏语法，该语法指定了 file，系指来源文档，code 便是要求要依照那个数据来计算过去的变量，var1 是指定要计算过去的变量，var2 是要求要命名的变量的前缀，n 则是要求要计算到过去第几个月的数据，抓取过去报酬率宏语法的优化写法如表 7-22 所示。这个宏语法虽然简便，如果细想规律，该语法是一次只针对一个过去数据做计算，亦即如果要取出过去 60 个月的

数据，则文件会被重复读取 60 次，该语法并不实用，该语法尚可修改如下。

表 7-22

<pre>%macro lag2(file,code,var1,var2,n); data &file; set &file; %do i=1 %to &n; &var2&i=lag&i(&var1); if &code^=lag&i(&code) then &var2&i=.; %end; run; %mend; %lag2(a,code,ret, r, 20);</pre>
--

该修正后的语法是将%do 与%end 向内缩至处理变量的部分，采用这样的写法，文档只会被读取一次，但是却可以直接处理 20 个变量，在程序的执行效率上会远优于前面的写法，而两个方式所得到的结果却是一模一样的。

我们同样可以将该语法应用在创造虚拟变量值，只要结合前一节的语法，我们就可以轻易建立公司、时间虚拟变量。因此，同样撰写两种语法，一种语法重复读取文档，另外一种则是只读取一次文档，读者可以自行比较两种语法撰写上的速度差异，两种虚拟变量宏语法的比较如表 7-23 所示。

表 7-23

<pre>proc sort data=a nodupkey out=b(keep=code) ;by code; run;</pre>	
<pre>%macro dummy1(file,code,dummy,n); %do i=1 %to &n; data &file; set &file ; &dummy&i=_n_=&i; run; %end; %mend; %dummy1(b,code,firm,1009);</pre>	<pre>%macro dummy2(file,code,dummy,n); data &file; set &file; %do i=1 %to &n; &dummy&i=_n_=&i; %end; Run; %mend; %dummy2(b,code,firm,1009);</pre>

在此处，我们产生了 1009 个虚拟变量，如果在后续要进行实证分析，就可以跟该虚拟变量的数据依照 code 合并即可，但在此处要特别提醒的是，由于公司虚拟变量相当多，在合并数据时会产生超过 1009 个变量，这将使得文档过于庞大，因此建议在回归分析前先行合并¹。

1 若是使用 SAS 内建的 panel data 以及 logit 程序，则有直接指定虚拟变量的语法，不需要再额外产生虚拟变量，但若只是进行 OLS 的话，则必须要自行先产生虚拟变量。

在前面的部分中，我们常常使用%do i=1 %to &n (或者一个确定的数字)，如果我们想要将相邻两个文件合并在一起会产生极大的困难，因为&i 会直接取代一个数字，SAS 内部提供了一个方法可提供作为计算使用，即采用宏语法生成数据集的语句如表 7-24 所示。

表 7-24

<pre>%macro a; %do i=1 %to 10; data data&i; do i=1 to 10; a=rannor(&i); output; end; run; %end; %mend a; %a;</pre>
--

首先我们先产生 10 个数据，其文档分别存为 a1、a2...a10，接下来我们要将相邻两个文档合并在一起，首先用以下的方法来解释，合并宏语法，如表 7-25 所示。

表 7-25

<pre>%macro aa; %do i=1 %to 9; data newa&i; merge data&i data&i+1; run; %end; %mend; %aa;</pre>

用 merge data&i data&i+1，SAS 真的无法将数据合并在一起吗？下面是程序执行的结果，错误的日志文件提示如图 7-9 所示。

```
127 %macro aa;
128 %do i=1 %to 9;
129 data newa&i;
130 merge data&i data&i+1;
131 run;
132 %end;
133 %mend;
134 %aa;
22: LINE and COLUMN cannot be determined.
NOTE 242-205: NOSPOOL is on. Rerunning with OPTION SPOOL may allow recovery of the LINE and COLUMN where the error has occurred.
ERROR 22-322: Syntax error, expecting one of the following: a name, a quoted string, (, -, :, ;, END, _DATA_, _LAST_, _NULL_.
76: LINE and COLUMN cannot be determined.
NOTE: NOSPOOL is on. Rerunning with OPTION SPOOL may allow recovery of the LINE and COLUMN where the error has occurred.
ERROR 76-322: Syntax error, statement will be ignored.
```

图 7-9

由结果可知，SAS 的确无法进行这样的计算，因此我们确实需要找出另外的方法来进行的语法，%eval 语法介绍如表 7-26 所示。

表 7-26

```
%MACRO aa;
%DO i=1 %to 9;
DATA NEW&i;
merge DATA&i DATA%eval(&i+1);
RUN;
%end;
%MEND;
%aa;
```

%eval 的语法就是能够计算&i 变量的宏函数，其出现情况仅能在已具有&i 指定的数字变量的情况下使用，如果单独存在就会产生错误。

7.3 宏语法撰写技巧

在前面两节，我们介绍了一些宏语法的撰写，在本节我们介绍如何将一般的语法修正为宏语法。下面先以 example_7_3.sas 为例介绍乐透抽样的语法（如表 7-27 所示），并且将该语法设置成一组套装化的可以使用的宏语法。

表 7-27

```
/* 乐透抽样语法*/
/*先撰写出一个完整可执行的 SAS 语法*/

data a;
do i=1 to 49;
output;
end;
run;

proc surveyselect data=a out=out noprint
seed=0 n=6 rep=10 method=srs;
run;

proc transpose data=out out=out (drop=_name_);
by replicate;
run;
```

该语法完成之后，每次只要执行完毕，就会抽出 10 张从 49 个号码中选 6 个号码的乐透，但是如果我们要抽的不是 10 张，而是 20 张；是 70 个号码抽 5 个号码而非 49 个号码选 6 个，那就必须要

重新修正了（如表 7-28 所示，将中国台湾地区乐透抽样语法改为宏语法），但是如果仔细思考，乐透抽样也只会更改这三个部分，而这样的语法一完成就是一组套装化的语法。

表 7-28

<pre>/*随便将撰写且可以执行的语法 塞到%macro a; %mend; 里面*/ %macro a; data a; do i=1 to 49; output; end; run; proc surveyselect data=a out=out noprint seed=0 n=6 rep=10 method=srs; run; proc transpose data=out out=out (drop=_name_); by replicate; run; %mend; %a;</pre>
--

在此处我们将语法塞到宏 a 里面，此后只要执行%a，就可以任意抽取出 10 张乐透彩票了，接下来我们再考虑一下，到底要抽 10 张还是要抽 20 张，或者是抽 15 张。在宏语法外围新增宏变量的语法如表 7-29 所示。

表 7-29

<pre>/*因为想要任意抽 10 组、20 组或者 15 组 这表示宏里面有一个变量要考虑 因此可以先新增变量*/ %macro a(num); data a; do i=1 to 49; output; end; run; proc surveyselect data=a out=out noprint seed=0 n=6 rep=10 method=srs; run; proc transpose data=out out=out (drop=_name_); by replicate; run; %mend; %a;</pre>

我们将 %macro a; 改成 %macro a(num); 但是在呼叫宏时，仍然撰写 %a，因为在宏语法中，尚未使用 num 变量，所以仍然可以顺利执行，接下来我们就要思考 num 这个变量应该放置在何处。将抽取次数改为宏变量的语法如表 7-30 所示。

表 7-30

```

/*将 rep=10 改成 rep=&num*/
%macro a(num);
data a;
do i=1 to 49;
output;
end;
run;
proc surveyselect data=a out=out noprint
seed=0 n=6 rep=&num method=srs;
/*因为是任意抽取几组，所以在此修改*/
run;
proc transpose data=out out=out (drop=_name_);
by replicate;
run;
%mend;
%a(10);
%a(20)
/*可以自由指定抽取几组乐透彩票了*/

```

虽然在程序的撰写教学上，我是采用先撰写 num 再将 rep=10 修改成 rep=&num 的，但是实际在程序呈现上，往往只会看到两个步骤合而为一的结果，接下来我们修改另外一个变量，也就是 49 个号码的语法，即将总号码个数改为宏变量，如表 7-31 所示。

表 7-31

```

%macro a(num,a);
data a;
do i=1 to &a;
/*修改要产生几个号码*/
output;
end;
run;
proc surveyselect data=a out=out noprint
seed=0 n=6 rep=&num method=srs;
run;

```

续表

```
proc transpose data=out out=out (drop=_name_);  
by replicate;  
run;  
%mend;  
%a(10,49);  
%a(10,70);
```

在表 7-31 中，我们新增了指定变量 a，再将 do i=1 to 49，改成 do i=1 to &a，在后续的语法中，就可以更改产生的号码数了，最后一个步骤，就是撰写抽出几个号码的语法了，将每张的号码数改为宏变量的语法如表 7-32 所示。

表 7-32

```
%macro a(num,a,b);  
data a;  
do i=1 to &a;  
output;  
end;  
run;  
proc surveyselect data=a out=out noprint  
seed=0 n=&b rep=&num method=srs;  
run;  
proc transpose data=out out=out (drop=_name_);  
by replicate;  
run;  
%mend;  
%a(10,49,6);  
%a(10,70,5);
```

到此为止，我们就完成了一个程序包的撰写了，而宏语法的功用就是将一些整理的方法套装化，特别是在进行实证中，有一些语法在撰写时会标准化的流程，不管用任何数据来撰写都是相同的步骤，则可以将其撰写成宏程序保存起来，之后在进行不同模型的实证时就可以马上使用。

在前一节中，我们撰写了%do i=1 %to 100；这样的语法，该语法类似于在 data set 中的 do i=1 to 100 的做法，而实质上，在 data set 中可以使用的条件句语法，都可在宏语法中撰写，只是在语法前面都要加上%才可以执行。在语法撰写上要用到%do i=1 %to 100，读者只需要考虑，该语法是针对变量处理，还是要针对文档处理，因为采用错误的处理方式，其最后所花费的时间将是天渊之别的。

以前一节的虚拟变量的设定而言，我们的目标是产生变量，如果可以的话，最好是不要重复读文档而用宏语法中的变量，文档与变量的宏语法优化比较程序如表 7-33 所示。

表 7-33

<pre>data a; do i=1 to 1000; output; end; run; %macro a; %do i=1 %to 1000; data a; set a; d&i=_n_=&i; run; %end; %mend;</pre>	<pre>data a; do i=1 to 1000; output; end; run; %macro a; data a; set a; %do i=1 %to 1000; d&i=_n_=&i; %end; run; %mend; %a;</pre>
---	---

这两者语法的差异，在于文档读取的次数，当我们将%do i=1 %to 1000 的语法放在 data set 之中，则可以预期只会读取一次文档，因此程序执行的速度会相当快，但是处理跨期的数据时，例如：moving window 与 rolling 的语法，此时就只能将%do i=1 %to 1000 放置在 data set 语法的外面了。

本节介绍简单的宏的撰写技巧以及需要注意的部分，读者在撰写宏时先撰写出可以执行的一般性的程序语法，再依序将相关变量替换掉，以及宏程序是要针对变量或者文档进行处理的，在撰写宏程序之前，只要能够掌握这两个原则，相信就能够写出丰富多彩的宏语法了。

7.4 总结

本章介绍了基础宏语法、进阶宏语法，基本上是提供一些撰写宏语法的逻辑和观念，宏语法的存在是需要大量重复执行程序的工作者的福音，要将宏程序撰写得好需要靠经验的累积，读者需要细心了解所有语法的一些规律性，及其基本观念，便可以撰写出可在日后重复使用的标准化的宏语法。

第 8 章

描述统计

在取得数据之后，需要对别人描述数据的特性，当然如果只有几笔数据，可以将所有的数据都详细说明，但在财务研究中，动辄百万笔的数据，便需要一个简单有效率的方式来告知他人数据特性。

本章介绍如何在 SAS 中适当地将数据整理成能说明重点的报表，以及介绍宏指令，主要是针对相关系数表格整理的语法，该语法在之后各章节皆会出现类似的指令，对于长期进行研究的人员将有很多的帮助。

8.1 常见的描述统计量

在有关描述统计量的程序语法中，SAS 提供了几个方便有效的程序，例如：proc means 以及 proc univariate。在功能上，proc means 与 proc univariate 都可以计算数据的集中量数（均值、中位数、四分位数、众数）、离散量数（全距、最大值、最小值、四分位距、方差、标准偏差）、偏态与峰态系数以及 t 统计量；其中 proc univariate 还提供单变量无母数检定、百分位数以及直方图等功能，功能比较强大；proc means 的功能虽然不强，但是在输出上较为简洁。本节介绍一些 proc means 以及 proc univariate 共同的指令，并介绍其异同，对于初学者而言，proc means 即可解决大部分的数据分析，但若是研究生或其他研究人员，则建议将 proc univariate 的语法了解得详尽些。两种描述统计程序的语法比较如表 8-1 所示。以下以 example_8_1.sas 为例来介绍有关描述统计的语法。

表 8-1

libname aa "D:\The Application of SAS in Financial Research\CH08\data";					
data a;					
set aa.sort;					
run;					
option nolabel;					
proc means data=a;					
var ret turn mv;					
run;					
proc univariate data=a;					
var ret turn mv;					
run;					

此处以第 4 章用来排序的数据来进行描述统计的语法介绍，分别采用 proc means 以及 proc univariate 来执行描述统计的功能。

使用 proc means 在 SAS 的 output 窗口的执行结果如图 8-1 所示，SAS 会制作简单的描述统计表格，并且报告各个变量的有效观测值（N）、均值（Mean）、标准偏差（Std Dev）、最小值（Minimum）以及最大值（Maximum）。

The MEANS Procedure					
Variable	N	Mean	Std Dev	Minimum	Maximum
ret	9201	0.2588349	14.3929810	-95.0000000	195.0000000
turn	9200	14.1397304	20.9319206	0	257.5600000
mv	9200	18542.30	72155.39	12.0000000	1545626.00

图 8-1

在 proc univariate 中，SAS 最详尽地将各个变量的相关数据都展现在 output 这个窗口里面，如图 8-2 所示。但是对用户而言，却稍嫌复杂一点，在 output 窗口中有动差 (moment)、基本观测值 (basic measure)、检定量 (tests for location)、分位数 (quantile)、极端值 (extreme ovservations) 以及遗漏值 (missing value)，尽管可以详尽地看到每个变量的所有重要数据，但却未必会是他们所关心的统计量。另一个描述统计功能的 proc means 是否就不够详尽了呢？此处以下列语法来展示如何个人化地使用 proc means。

The UNIVARIATE Procedure			
Variable: ret			
Moments			
N	9201	Sum Weights	9201
Mean	0.25883491	Sum Observations	2381.54
Std Deviation	14.392381	Variance	207.157303
Skewness	1.25780559	Kurtosis	14.1601051
Uncorrected SS	1908469.13	Corrected SS	1905852.7
Coeff Variation	5560.68	Std Error Mean	0.15004905
Basic Statistical Measures			
Location		Variability	
Mean	0.25883	Std Deviation	14.39238
Median	-0.47000	Variance	207.15730
Mode	0.00000	Range	290.00000
		Interquartile Range	12.14000
Tests for Location: Mu0=0			
Test	-Statistic-	-----p Value-----	
Student's t	t 1.725002	Pr > t	0.0846
Sign	M -225	Pr >= M	<.0001
Signed Rank	S -788573	Pr >= S	0.0014
Quantiles (Definition 5)			
Quantile		Estimate	
100% Max		195.00	
99%		46.24	
95%		22.22	
90%		14.69	
75% Q3		5.71	
50% Median		-0.47	
25% Q1		-6.43	
10%		-13.28	
5%		-18.75	
1%		-36.65	
0% Min		-95.00	
Extreme Observations			
-----Lowest-----		-----Highest-----	
Value	Obs	Value	Obs
-95.00	8434	105.88	7624
-85.11	8882	114.88	9149
-80.87	8448	138.10	9021
-79.40	9190	173.97	8476
-79.21	8854	195.00	9109

图 8-2

只要读者查询 SAS 中的 help 去搜寻 means 这个 procedure，就可以找出 proc means 提供的所有统计量语句，如表 8-2 所示，在撰写程序时，可以要求 SAS 在 output 中展示哪些统计量¹，而非内建的 N、Mean、Std Dev、Minimum 以及 Maximum，而且在数据的展现上，是将所有想探讨的变量放置在一起，可以一次就了解 ret、turn 以及 mv 的均值、标准偏差等各个统计量的数据，proc means 详尽统计量的

1 Means 内建的统计量共有 29 个。

执行结果如图 8-3 所示。虽然其缺点是要自行撰写相关统计量的输出，但在可读性方面则优于 proc univariate 的 output 报表。

表 8-2

```
proc means data=a n mean t probt max p99 p95 p90  q3 median  q1 p10 p5 p1 min  std stderr skewness kurtosis ;
var ret turn mv;
run;
```

The MEANS Procedure									
Variable	N	Mean	t Value	Pr > t	Maximum	99th Pctl	95th Pctl	90th Pctl	Upper Quartile
ret	9201	0.2588349	1.73	0.0846	195.0000000	46.2400000	22.2200000	14.6900000	5.7100000
turn	9200	14.1397304	64.79	<.0001	257.5600000	103.1700000	56.9350000	37.2800000	16.7000000
mv	9200	18542.30	24.65	<.0001	1545626.00	299269.00	69422.00	31149.50	8847.00

Variable	Median	Lower Quartile	10th Pctl	5th Pctl	1st Pctl	Minimum	Std Dev	Std Error
ret	-0.4700000	-6.4300000	-13.2800000	-18.7500000	-36.6500000	-95.0000000	14.3929810	0.1500490
turn	6.3400000	2.2600000	0.5700000	0.0900000	0.0100000	0	20.9319206	0.2182304
mv	3263.00	1255.50	474.0000000	224.0000000	77.0000000	12.0000000	72155.39	752.2719628

Variable	Skewness	Kurtosis
ret	1.2578056	14.1601051
turn	3.2807701	15.8245946
mv	10.7522667	159.9728574

图 8-3

虽然 SAS 在 output 窗口上输出的报表能够相当清楚地展示，但学术研究者更关心的是能否将该结果复制粘贴到 Word 文档上呢？如果粘贴上最原始的 proc means 结果，则发现直接将 output 的报表复制是不可用的，因此必须再经过特殊的处理才行，那么是否能将描述统计量的结果存到 SAS 的 table 呢？如果可行的话，就能够将其快速地输出到 office 的 Excel 文档中，那么整理表格就很方便了。将描述统计量输出到 SAS 文档的程序如表 8-3 所示。

The MEANS Procedure				
		Variable	N	Mean
Minimum	Maximum			Std Dev

		ret	9201	0.2588349
-95.0000000	195.0000000			14.3929810
		turn	9200	14.1397304
0	257.5600000			20.9319206
		mv	9200	18542.30
12.0000000	1545626.00			72155.39

表 8-3

```
proc means data=a noprint;
var ret turn mv;
output out=b;
run;
```

在上述程序中，利用 proc means 可以快速地输出简单的表格，如图 8-4 所示。

	TYPE	_FREQ_	_STAT_	ret	turn	mv
1	0	9201	N	9201	9200	9200
2	0	9201	MIN	-95	0	12
3	0	9201	MAX	195	257.56	1545626
4	0	9201	MEAN	0.2588349092	14.139730435	18542.295326
5	0	9201	STD	14.392981019	20.931920578	72155.39187

图 8-4

这样，在输出时可以轻易地输出 proc means 的内建表格，然而在 proc univariate 则缺乏这项功能，如果希望描述统计量为变量，ret、turn、mv 为观测值，则可以利用转置功能，输出结果转置整理的语句如表 8-4 所示。数据转置的结果如图 8-5 所示。

表 8-4

```
proc transpose data=b out=c;
var ret turn mv;
id _stat_;
run;
```

	NAME	N	MIN	MAX	MEAN	STD
1	ret	9201	-95	195	0.2588349092	14.392981019
2	turn	9200	0	257.56	14.139730435	20.931920578
3	mv	9200	12	1545626	18542.295326	72155.39187

图 8-5

虽然 proc univariate 不能输出内建的基本 5 个统计量，但和 proc means 一样，都具备输出特定统计量的功能，完整的描述统计量的语法如表 8-5 所示，其输出方法有很多，本章节介绍最能够个人化的程序撰写方法。

表 8-5

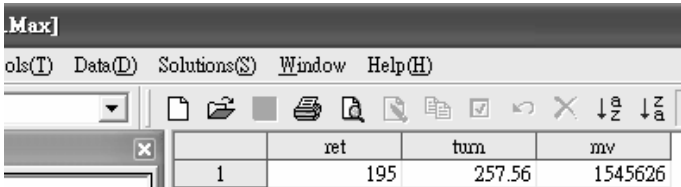
```
proc univariate data=a noprint; /*proc means 亦可执行*/
var ret turn mv;
output out=n n= ret turn mv;;
```

续表

```
Output out=mean mean= ret turn mv;
output out=t t= ret turn mv;
output out=probt probt= ret turn mv;
output out=max max= ret turn mv;
output out=p99 p99= ret turn mv;
output out=p95 p95= ret turn mv;
output out=p90 p90= ret turn mv;
output out=q3 q3= ret turn mv;
output out=median median= ret turn mv;
output out=q1 q1= ret turn mv;
output out=p10 p10= ret turn mv;
output out=p5 p5= ret turn mv;
output out=p1 p1= ret turn mv;
output out=min min= ret turn mv;
output out=var var=ret turn mv;
output out=std std= ret turn mv;
output out=stderr stderr= ret turn mv;
output out=skew skewness= ret turn mv;
output out=kurt kurtosis= ret turn mv;
```

run;

以上程序会分别将各个描述统计量输出到以统计量命名的报表中，以最大值为例，输出报表如图 8-6 所示。

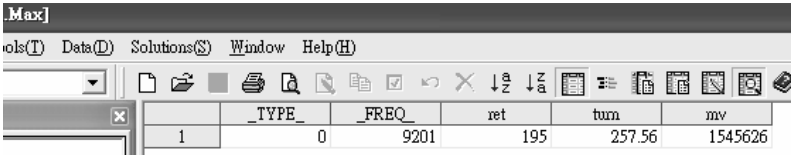


The screenshot shows the SAS Max window with the menu bar (ols(T), Data(D), Solutions(S), Window, Help(H)) and a toolbar. Below the toolbar is a table with the following data:

	ret	turn	mv
1	195	257.56	1545626

图 8-6

如果使用 proc means 来执行这个程序，结果则会多出两个变量，如图 8-7 所示。



The screenshot shows the SAS Max window with the menu bar (ols(T), Data(D), Solutions(S), Window, Help(H)) and a toolbar. Below the toolbar is a table with the following data:

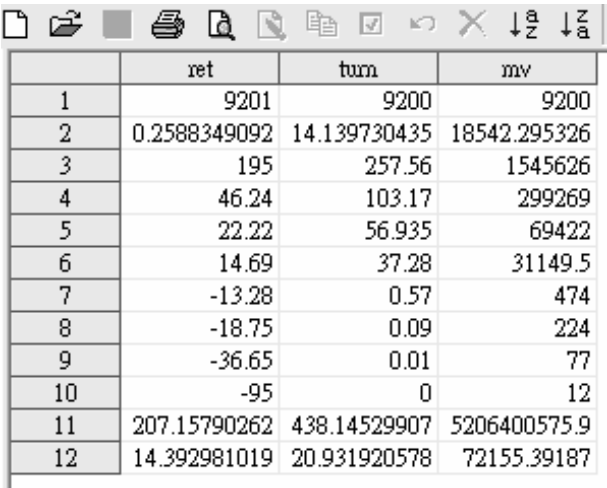
	TYPE	_FREQ_	ret	turn	mv
1	0	9201	195	257.56	1545626

图 8-7

此处采用如表 8-6 所示的执行结果合并语法，将几个感兴趣的变量垂直合并在一起时，语法合并的结果如图 8-8 所示。

表 8-6

```
data final;
set n mean max p99 p95 p90 p10 p5 p1 min var std;
run;
```



	ret	tum	mv
1	9201	9200	9200
2	0.2588349092	14.139730435	18542.295326
3	195	257.56	1545626
4	46.24	103.17	299269
5	22.22	56.935	69422
6	14.69	37.28	31149.5
7	-13.28	0.57	474
8	-18.75	0.09	224
9	-36.65	0.01	77
10	-95	0	12
11	207.15790262	438.14529907	5206400575.9
12	14.392981019	20.931920578	72155.39187

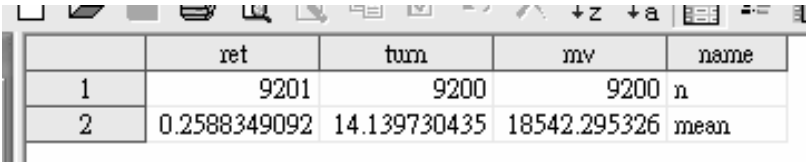
图 8-8

虽然数据是整理出来了，但是结果却不佳，因为用户无法确定各个观测值属于哪一个统计量，此时可以采取如表 8-7 所示的描述统计量数据的再整理语法的方法来加以处理。

表 8-7

```
data n;
set n;
    name="n";
run;
data mean;
set mean;
    name="mean";
run;
data final;
set n mean ;
run;
```


此处用户可以利用自行命名的方式，重新创建一个 name 的变量，并且为这个变量命名为描述统计量的名称，由于变量为文本文档，因此命名上要以两个引号夹住名称，本例中以两个描述统计量为例，描述统计量再整理结果的如图 8-8 所示。若要多一点统计量，则需要将每个文档都整理一次，最后再进行合并。



	ret	turn	mv	name
1	9201	9200	9200	n
2	0.2588349092	14.139730435	18542.295326	mean

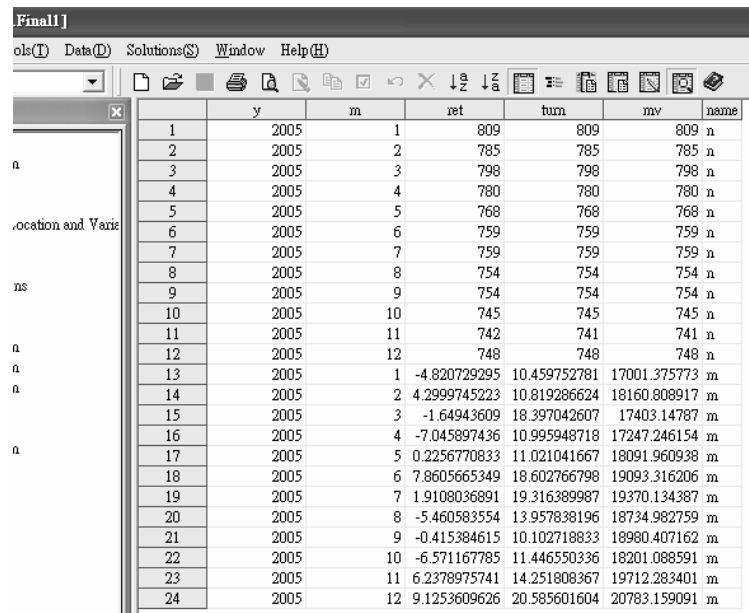
图 8-8

依特定变量进行描述统计的语句如表 8-8 所示。

表 8-8

```
proc sort data=a;by y m;
run;
proc univariate data=a noprint; /*proc means 亦可执行*/
var ret turn mv;
output out=n n= ret turn mv;;
output out=mean mean= ret turn mv;
by y m;
run;
data n;
set n;
    name="n";
run;
data mean;
set mean;
    name="mean";
run;
data final;
set n mean ;
run;
```

采用上述描述统计语句时，我们将数据依照月份排序，执行后的结果如图 8-9 所示。但由此图可见，将会使数据每个月都要计算一次，如此是否会影响合并文档的结果？



	y	m	ret	turn	mv	name
	1	2005	1	809	809	809 n
	2	2005	2	785	785	785 n
	3	2005	3	798	798	798 n
	4	2005	4	780	780	780 n
	5	2005	5	768	768	768 n
	6	2005	6	759	759	759 n
	7	2005	7	759	759	759 n
	8	2005	8	754	754	754 n
	9	2005	9	754	754	754 n
	10	2005	10	745	745	745 n
	11	2005	11	742	741	741 n
	12	2005	12	748	748	748 n
	13	2005	1	-4.820729295	10.459752781	17001.375773 m
	14	2005	2	4.2999745223	10.819286624	18160.808917 m
	15	2005	3	-1.64943609	18.397042607	17403.14787 m
	16	2005	4	-7.045897436	10.995948718	17247.246154 m
	17	2005	5	0.2256770833	11.021041667	18091.960938 m
	18	2005	6	7.8605665349	18.602766798	19093.316206 m
	19	2005	7	1.9108036891	19.316389987	19370.134387 m
	20	2005	8	-5.460583554	13.957838196	18734.982759 m
	21	2005	9	-0.415384615	10.102718833	18980.407162 m
	22	2005	10	-6.571167785	11.446550336	18201.088591 m
	23	2005	11	6.2378975741	14.251808367	19712.283401 m
	24	2005	12	9.1253609626	20.585601604	20783.159091 m

图 8-9

结果发现，不管是否要分组计算描述统计量，使用者依然可以利用如表 8-9 所示的描述统计量输出语句进行个人化的统计量计算，并且能够顺利地存储成 SAS 文档，再利用第 1 章输出到 Excel 文档的功能，就能顺利地将表格粘贴到 Word 文档中。

表 8-9

```
proc export data=final
  outfile="D:\The Application of SAS in Financial Research\CH08\output\描述统计"
  dbms=xlsx
  replace;
  sheet="个人化";
run;
```

前面的部分是计算简单平均的做法，在 SAS 中还可计算加权平均，如在计算股票报酬率时，经常要计算市值加权平均报酬，接下来便介绍加权平均的语法，如表 8-10 所示。

表 8-10

```
proc univariate data=a ; /*proc means 亦可执行*/
  var ret ;
  weight mv;
  by y m;
```

续表

<pre>run; proc means data=a; var ret; weight mv; by y m; run;</pre>

proc means 以及 proc univariate 共同的语法介绍，如表 8-11 所示。

表 8-11

语 法	含 义
Var	针对后续变量计算其统计量
By	针对组别计算统计量，在此之前，必须要先排序
Weight	然后对变量计算加权平均
Noprint	要求 SAS 不要在 output 窗口 展示结果
output out=文档名 统计量=变量名	文档名可以任意命名，常见的统计量有 n mean t probt max p99 p95 p90 q3 median q1 p10 p5 p1 min std stderr skewness kurtosis，除此之外，使用者还可在 help 文档中查询其他重要的统计量

8.2 相关系数

在大数据时代中，有时候不仅仅要取得庞大的样本数据，也会同时取得多个变量的数据，此时研究者不但关心单一变量的描述统计量，也关心两个变量间的关系。实际上，相关系数就是用来衡量两个变量同时上升以及同时下降的程度，若 x1 与 x2 的相关系数为 0.9，其解释为若观察到 x1 增加 1 单位时，也会看到 x2 同时增加 0.9 单位，但其不能描述变量间因果关系，这是在使用相关系数时应特别注意的地方。

SAS 提供了 proc corr 这个程序来计算变量间的相关系数，例如可计算皮尔森相关系数（pearson correlation coefficient）以及斯皮尔曼等级相关系数（spearman rank correlation coefficient）等，SAS 还可提供其他种类相关系数的运算，但在一般数据分析里，这两种相关系数已经足够说明数据的特性，本节介绍这两种相关系数的程序语法，而协方差是计算皮尔森相关系数的重要统计量，SAS 也能提供如何输出协方差矩阵的方法，这点在财务领域中是相当重要的，特别是在进行多元化投资组合时，其风险就是由报酬率的协方差矩阵计算而来的，以下以 example_8_2.sas 为例来进行介绍。

计算相关系数程序的步骤，如表 8-12 所示。

表 8-12

```
libname aa "D:\The Application of SAS in Financial Research\CH08\data";  
data a;  
set aa.sort;  
  
run;  
  
proc corr data=a;  
var ret turn mv;  
  
run;
```

首先第一步先介绍不针对 proc corr 做额外设定的程序语法,然后检查该结果在 output 窗口的结果。计算相关系数程序步骤的执行结果如图 8-10 所示。

The CORR Procedure						
3 Variables: ret turn mv						
Simple Statistics						
Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
ret	9201	0.25883	14.39298	2382	-95.00000	195.00000
turn	9200	14.13973	20.93192	130086	0	257.56000
mv	9200	18542	72155	170588117	12.00000	1545626
Pearson Correlation Coefficients						
Prob > r under H0: Rho=0						
Number of Observations						
		ret	turn	mv		
ret		1.00000	0.27205	0.02547		
			<.0001	0.0146		
	9201		9200	9200		
turn		0.27205	1.00000	-0.03861		
		<.0001		0.0002		
	9200		9200	9200		
mv		0.02547	-0.03861	1.00000		
		0.0146	0.0002			
	9200		9200	9200		

图 8-10

在使用 SAS 的默认功能时,会输出简单的描述统计量与皮尔森相关系数,如果要求输出斯皮尔曼等级相关系数,或者同时输出两种以上的相关系数,则可以使用如表 8-13 所示的程序语法。皮尔森相关系数与斯皮尔曼等级相关系数的执行结果如图 8-11 所示。

表 8-13

```
proc corr data=a pearson spearman;  
var ret turn mv;  
  
run;
```

The CORR Procedure						
3 Variables: ret turn mv						
Simple Statistics						
Variable	N	Mean	Std Dev	Median	Minimum	Maximum
ret	9201	0.25883	14.39298	-0.47000	-95.00000	195.00000
turn	9200	14.13973	20.93192	6.34000	0	257.56000
mv	9200	18542	72155	3263	12.00000	1545626
Pearson Correlation Coefficients						
Prob > r under H0: Rho=0						
Number of Observations						
		ret	turn	mv		
ret		1.00000	0.27205	0.02547		
			<.0001	0.0146		
	9201		9200	9200		
turn		0.27205	1.00000	-0.03861		
		<.0001		0.0002		
	9200		9200	9200		
mv		0.02547	-0.03861	1.00000		
		0.0146	0.0002			
	9200		9200	9200		
Spearman Correlation Coefficients						
Prob > r under H0: Rho=0						
Number of Observations						
		ret	turn	mv		
ret		1.00000	0.23141	0.14887		
			<.0001	<.0001		
	9201		9200	9200		

图 8-11

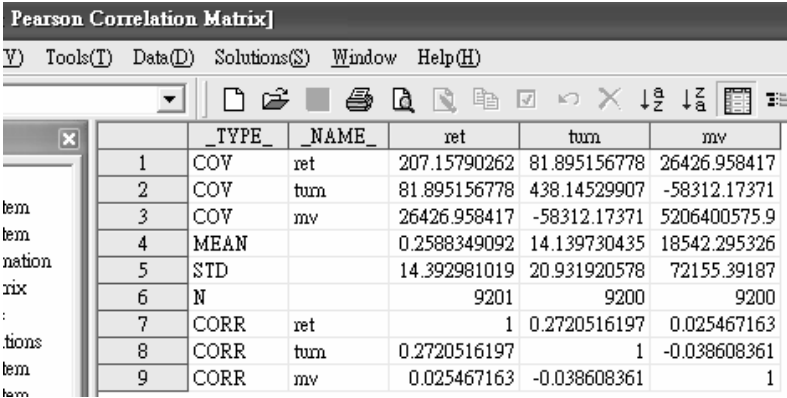
由结果来看是挺清楚的，但是用户无法将其顺利地整理成表格，接下来，介绍如何将数据存储成 SAS 的 table 文档。相关系数结果存为 SAS 文档的语句如表 8-14 所示。

表 8-14

```
proc corr data=a outp=p outs=s cov noprint;
var ret turn mv;
run;
```

采用该程序语法，我们就能将皮尔森相关系数存储成 p，斯皮尔曼等级相关系数存储成 s，由于加了 cov 这个选项语法，SAS 会将协方差矩阵也一并输出到 p。而这个协方差矩阵，将会是投资组合章节中一个关键核心的角色，利用协方差矩阵可以画出多元化投资风险的极限，利用协方差矩阵亦可算

出效率投资组合的权重以及绘出图形。
皮尔森相关系数的 SAS 表格如图 8-12 所示。

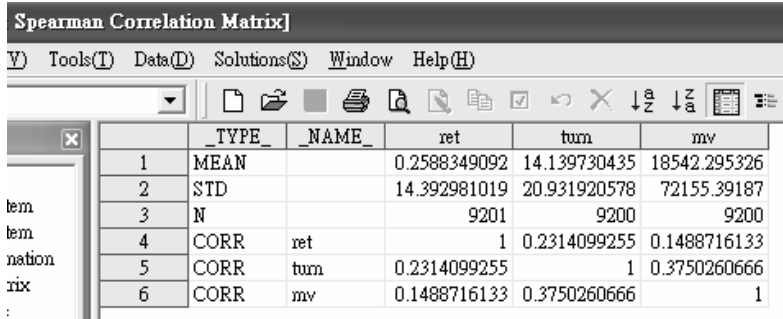


The screenshot shows the SAS 'Pearson Correlation Matrix' window. It contains a table with 9 rows and 6 columns. The columns are labeled '_TYPE_', '_NAME_', 'ret', 'turn', and 'mv'. The rows represent different statistical measures: 1. COV for 'ret', 2. COV for 'turn', 3. COV for 'mv', 4. MEAN, 5. STD, 6. N, 7. CORR for 'ret', 8. CORR for 'turn', and 9. CORR for 'mv'. The table also includes summary statistics like means, standard deviations, and sample sizes for each variable.

	TYPE	_NAME_	ret	turn	mv
1	COV	ret	207.15790262	81.895156778	26426.958417
2	COV	turn	81.895156778	438.14529907	-58312.17371
3	COV	mv	26426.958417	-58312.17371	5206400575.9
4	MEAN		0.2588349092	14.139730435	18542.295326
5	STD		14.392981019	20.931920578	72155.39187
6	N		9201	9200	9200
7	CORR	ret	1	0.2720516197	0.025467163
8	CORR	turn	0.2720516197	1	-0.038608361
9	CORR	mv	0.025467163	-0.038608361	1

图 8-12

在皮尔森相关系数的 table 中可以看到协方差矩阵、均值、标准偏差、有效观测值以及相关系数表。
斯皮尔曼等级相关系数的 SAS 表格如图 8-13 所示。



The screenshot shows the SAS 'Spearman Correlation Matrix' window. It contains a table with 6 rows and 6 columns. The columns are labeled '_TYPE_', '_NAME_', 'ret', 'turn', and 'mv'. The rows represent different statistical measures: 1. MEAN, 2. STD, 3. N, 4. CORR for 'ret', 5. CORR for 'turn', and 6. CORR for 'mv'. The table also includes summary statistics like means, standard deviations, and sample sizes for each variable.

	TYPE	_NAME_	ret	turn	mv
1	MEAN		0.2588349092	14.139730435	18542.295326
2	STD		14.392981019	20.931920578	72155.39187
3	N		9201	9200	9200
4	CORR	ret	1	0.2314099255	0.1488716133
5	CORR	turn	0.2314099255	1	0.3750260666
6	CORR	mv	0.1488716133	0.3750260666	1

图 8-13

在斯皮尔曼等级相关系数的 table 中则看不到协方差矩阵，虽然利用上述的指令可以将相关系数结果输出到 table 里，但是缺少了显示的 p-value，而在一般数据分析中，也很少会将相关系数的 p-value 报告出来，而是标注显示星号，那么 SAS 是否具有这样的功能呢？利用 SAS 的 Output Delivery System (ODS) 的功能，可以取得相关的数据。

在此处先简单介绍一下 ODS 这个系统语法，由名称中可以了解 ODS 是针对 SAS output 窗口进行的传送语法，SAS 可以将输出到 output 窗口的数据转成 SAS 的 table，因此 ODS 是存在于不同的程序中的，例如，proc univariate 有相关的 ODS 语法，但是 proc means 则没有相关 ODS 的语法，用户可以在 SAS 中查询某个程序步是否有 ODS 的功能，ODS 的使用必须用试错法才能找出自己想要的 table，在本书中出现的 ODS 语法皆是笔者尝试过后最常使用的表格，但使用 ODS 功能有一个限制，亦即无法使用 noprint 这个功能，因为 noprint 是抑制 SAS 将结果展示到 output 窗口，而 ODS 是要求 SAS 将

output 窗口的结果传送到 SAS 的 table 文档，如果抑制了 output 的结果，那么 ODS 的功能也会自动失效。

相关系数语句中 ODS 的运用如表 8-15 所示。相关系数语句的三大 ODS 表格如图 8-14 所示。

表 8-15

```
proc corr data=a spearman cov outp=cov;  
var ret turn mv;  
ods output pearsoncorr=pear;  
ods output simplestats=d;  
ods output spearmancorr=spear;  
run;
```

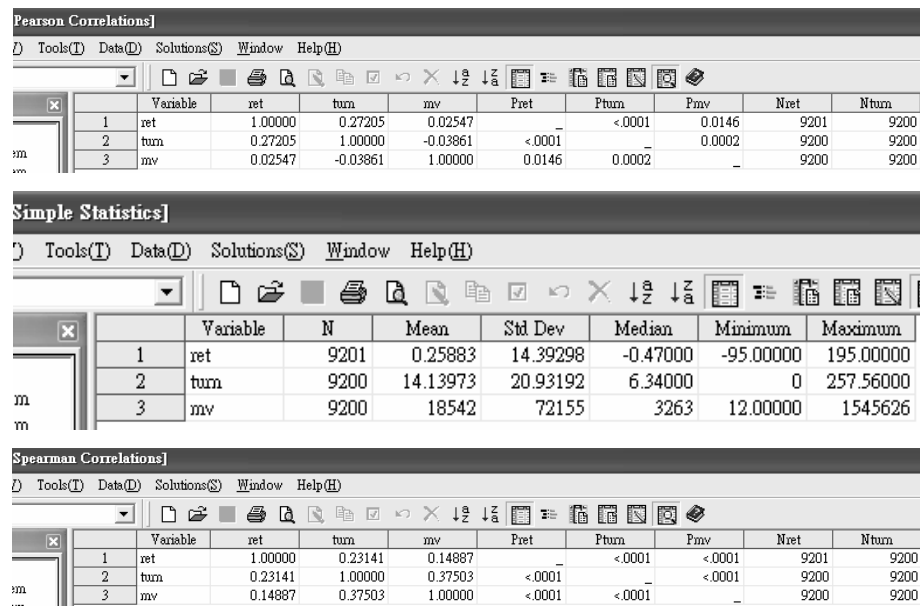


图 8-14

首先先检验 c、d、e 这三个 table。

仔细观察输出的结果，发现除了简单统计量的结果外，相关系数表格恐怕需要进行进一步的整理，这部分的程序将会使用宏（%macro）的语法，我们在 8.3 节会详细介绍撰写宏程序的方法，并且在稍后的章节会陆续介绍笔者自行创建的宏程序，在进行数据分析时，可以省去将 output 窗口的表格转成正式报告用形式的时间。

相关系数程序步的介绍如表 8-16 所示。

表 8-16

语 法	含 义
Outp	将皮尔森相关系数输出到 table，若有增列 cov 这个选项，则会输出协方差矩阵
Outs	将斯皮尔曼等级相关系数输出到 table
Pearson	要求 SAS 将皮尔森相关系数输出到 output 窗口
Spearman	要求 SAS 将斯皮尔曼等级相关系数输出到 output 窗口
Cov	要求 SAS 将协方差矩阵输出到 output 窗口，若同时下 outp 以及 cov 指令，协方差矩阵也会输出到 outp 文档中
Var	针对变量计算其相关系数
By	依据特定组别计算相关系数
Noprint	要求 SAS 不要在 output 窗口展示结果，但若要是使用 ODS 功能则不可使用这个指令
Ods output	以下为 proc corr 可用的输出表格
Pearsoncorr	要求 SAS 将皮尔森相关系数的 output 窗口的结果输出到 SAS 的 table，为默认值
Spearmancorr	要求 SAS 将斯皮尔曼等级相关系数的 output 窗口的结果输出到 SAS 的 table，但先前必须下 spearman 这个指令
Simplestats	要求 SAS 将简单统计量的 output 窗口的结果输出到 SAS 的 table，为默认值

8.3 个人化表格宏解析

这部分我们简单介绍笔者进行实证时采用的相关系数宏语法，由于本章节使用的学术型宏语法会重复出现在之后的各章节中，主要是为了将数据表格整理成财务实证论文使用的形式，若对该宏语法有浓厚兴趣的人，可以自行将程序设计重新编码，以成为读者个人化的语法。corr.sas 这个宏文档的介绍如表 8-17 所示，特别要注意，该文档使用的是双重宏。

表 8-17

<pre>%macro correlation(file,bit,varlist); proc corr data=&file spearman cov outp=cov; var &varlist; ods output pearsoncorr=pear; ods output simplestats=d; ods output spearmancorr=spear; run; %macro corr (file,file2); data d; set d nobs=x; call symput('de',x);</pre>
--

续表

```

run;
%do j=1 %to &de;
  data _null_;
    set &file;
    if _n_=&j then do;
      call symput('ss', variable);
    end;
run;
data a&j;
  set &file;
  b=round(&ss,10**(-1*&bit));
  if 0<p&ss<0.01 then c='***';
  if 0.01<=p&ss<0.05 then c='**';
  if 0.05<=p&ss<0.1 then c='*';
  if 0.1<=p&ss then c='';
  a&j=b||c;
  if _n_<&j then a&j='';
  keep variable a&j;
run;
data a&j;
  set a&j;
  &ss=a&j;
  keep variable &ss ;
run;
%end;
data &file2;
set a1;
run;
%do k=2 %to &de;
  data &file2;
    merge &file2 a&k;
  run;
%end;
%mend;
%corr (pear,pearson);
%corr (spear,spearman);
proc delete data= d pear spear;
quit;
%mend;

```

程序执行后的结果，如图 8-15 所示。

Pearson				
ols(T) Data(D) Solutions(S) Window Help(H)				
	Variable	ret	tum	mv
1	ret	1		
2	tum	0.272***	1	
3	mv	0.025**	-0.039***	1

Spearman				
ols(T) Data(D) Solutions(S) Window Help(H)				
	Variable	ret	tum	mv
1	ret	1		
2	tum	0.231***	1	
3	mv	0.149***	0.375***	1

图 8-15

虽然该宏能够将皮尔森相关系数以及斯皮尔曼等级相关系数的表格做个人化的设计输出。然而，该程序语法有点长，即使是在日常使用中采用复制粘贴的方式，也相当不便，笔者建议采取以下的步骤，可以将执行程序写得更为简洁一点。

本书已经将事先设计好的宏程序存成文件名为 corr, sas 的程序文件，使用相关系数分析时先找出 corr, sas 所在文档的路径。相关系数宏程序的文件如图 8-16 所示。

D:\The Application of SAS in Financial Research\CH08\program		
包含到库中 共享 刻录 新建文件夹		
名称	修改日期	
corr	2010/3/29 9:55	
example_8_1	2015/6/17 16:11	
example_8_2	2015/6/17 16:14	
example_8_3	2015/6/17 16:15	
example_8_4	2009/11/25 2:35	

图 8-16

找出该宏语法路径后，以 example_8_3.sas 来介绍如何使用相关系数以及描述统计表的宏语法，在之后的章节中，本书提供不同的宏语法，都是使用读取外部宏文档的方式来运行的，这样可节省读者进行数据分析的时间，相关系数与描述统计外部宏执行的语句如表 8-18 所示。

表 8-18

libname aa "D:\The Application of SAS in Financial Research\CH08\data";	
data a;	
set aa.sort;	
run;	
%include "D:\The Application of SAS in Financial Research\CH08\program_cn\corr.sas";	
%correlation(a,4,ret mv turn);	
/*	
a 为要检定相关系数表的文档	
4 为小数点位数	
ret mv turn 为要检定的变量	
会产生 pearson 以及 spearman 两个文档	
*/	
%inc 'D:\The Application of SAS in Financial Research\CH08\program_cn\stat.sas';	
%stat(a,4,ret mv turn);	
/*	
a 可改为要检定的文件	
4 为小数点位数	
ret mv turn 为要计算描述统计量的变量	
会产生 final final1 两个文件*/	

在完成了宏语法后，读者只需要读入已撰写好的宏语法的文档，就可以用相当简洁的方式进行描述统计表或者相关系数表了，其中读入已撰写好的 SAS 语句通常是采用%include 或者%inc，如 8-19 所示。

表 8-19

语 法	含 义
%include	呼叫外部的 SAS 文档，通常用来呼叫复杂的宏程序

在前面曾经提过宏程序是专门用来给不同文档执行相同的程序所用的，而如果宏程序撰写的方式符合标准化过程，其宏程序就可以在不同的数据中重复使用，而用户也可以在网络上搜索别人已经撰写好的宏程序进行修改，就可以便利地重复使用其程序。

8.4 趋势图基础语法介绍

在描述统计或者一些研究报告中，图形是一个相当重要的报告方式，利用呈现的图形，读者可以轻易地了解结果，下面以 example_8_4 为例来介绍绘图数据整理的语法，如表 8-20 所示。

表 8-20

```
libname aa "D:\The Application of SAS in Financial Research\SAS data\monthly price";  
data a;  
set aa.price;  
month=y*100+m;  
date=mdy(m,1,y);  
format date yymon6.;  
if y<2000 then delete;  
if code=1101 or code=1102 or code=2002 or code=2006 or code=2303 or code=2330  
or code=2608 or code=2609 then output;  
run;
```

为了绘图的方便，我们先从股价数据中挑选出 8 只股票作为绘图介绍，虽然 SAS 常被批评为绘图功能不佳，但是在 SAS 9.2 版以后，已经大幅度地强化了其绘图功能，其中以 proc sgpanel 以及 proc sgplot 为主，可以为使用者快速地画出漂亮的图形，以下分别以 proc sgpanel 与 proc sgplot 两种程序进行介绍。Sgpanel 程序步介绍如表 8-21 所示。

表 8-21

```
proc sgpanel data=a;  
title "The pattern of return";  
panelby code ;  
series x=month y=ret;  
run;
```

首先，画出一个时间序列图，在该语法中，可以先指定图表的 title，接着指定依照 code 来分别画图，由于此处要绘制时间序列图形，所以使用 series 就可以画出趋势图，接着指定 x 轴的变量与 y 轴的变量，就可以画出如图 8-17 所示的图形。

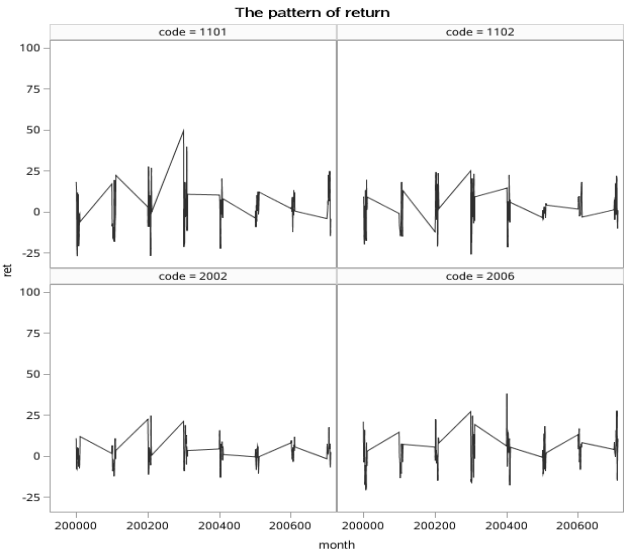


图 8-17

就图形来看会产生极大的问题，因为在此处使用的 month 变量是 $y*100+m$ ，虽然就阅读上，用户知道 200001 是 2000 年 1 月，但是对 SAS 而言则是 20 万零 1 的意思，因此有必要采用时间格式的数据来进行绘图，以时间变量执行 sgpanel 的语句如表 8-22 所示。

表 8-22

<pre>proc sgpanel data=a; title "The pattern of return"; panelby code ; series x=date y=ret; run;</pre>

以时间变量执行 sgpanel 的结果如图 8-18 所示。

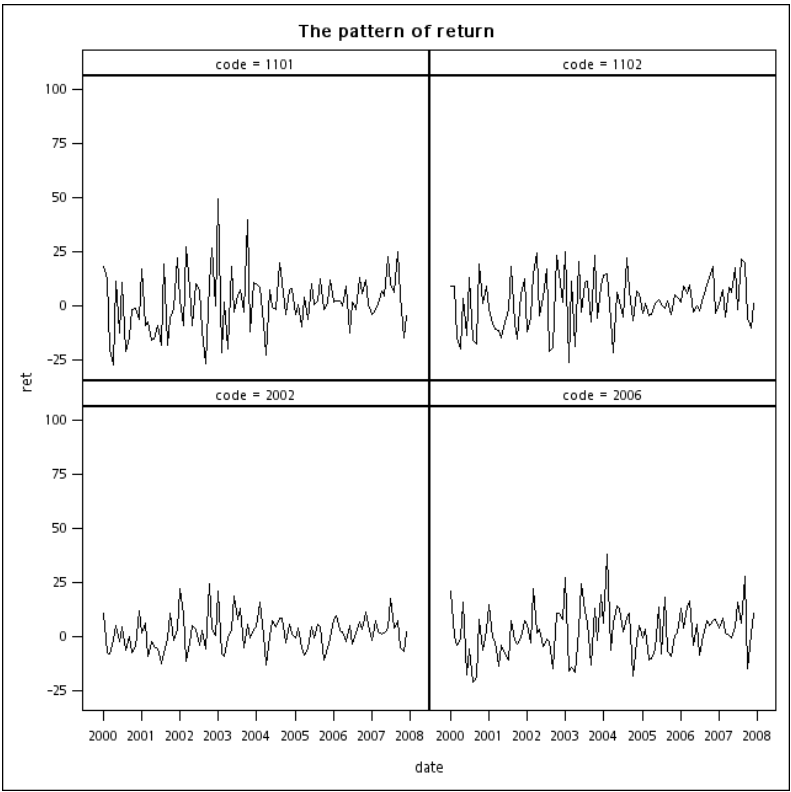


图 8-18

如此即可将数据绘成漂亮的时间趋势图，而采用 proc sgpanel 时，预设的绘图方式是 2 乘 2 的表格呈现方式，依此仅能看到四家公司，如果认为图形太小，则可以用如表 8-23 所示的语句来进行更改输出的图例数。

表 8-23

```
proc sgpanel data=a;  
title "The pattern of return";  
panelby code /columns=4;  
series x=date y=ret;  
run;
```

更改输出图例数的执行结果如图 8-19 所示。

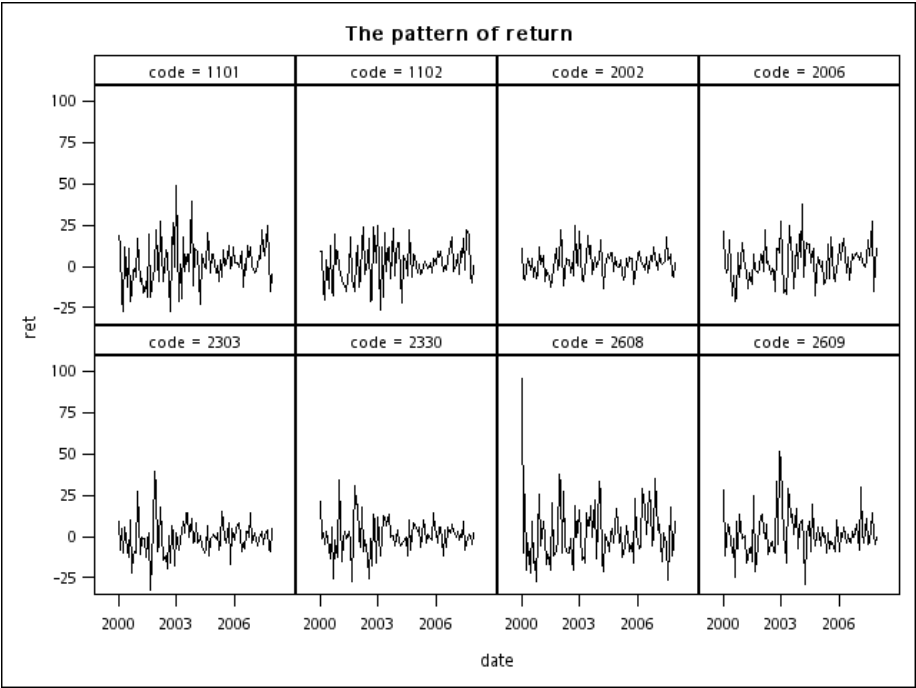


图 8-19

由表 8-23 可知，只要在 panelby code 后面加一个/columns=4，就可以将图形变为 4 乘 2 的图示，当然也可以将图形改为 2 乘 4 或者 3 乘 3 的样式，3 乘 3 的作图格式语句如表 8-24 所示。

表 8-24

```
proc sgpanel data=a;  
title "The pattern of return";  
panelby code /columns=3 rows=3;  
series x=date y=ret;  
run;
```

由表 8-24 所示的语法可以很清楚地看到，如果用户要指定直行与横列的图形个数，可以直接撰写，在这里要提供一个经验，在图形的呈现中，columns 的个数最好要大于 rows 的个数，这在视觉上以及文档的编辑上比较好。

有时候，想同时画出几种数据的序列作比较，在 proc sgpanel 中，可以用如表 8-25 所示的语句进行撰写，多数列的输出结果如图 8-20 所示。

表 8-25

```
proc sgpanel data=a;
title "The pattern of return vs turnover";
panelby code / columns=3 rows=3;
series x=date y=ret;
series x=date y=turn;
run;
```

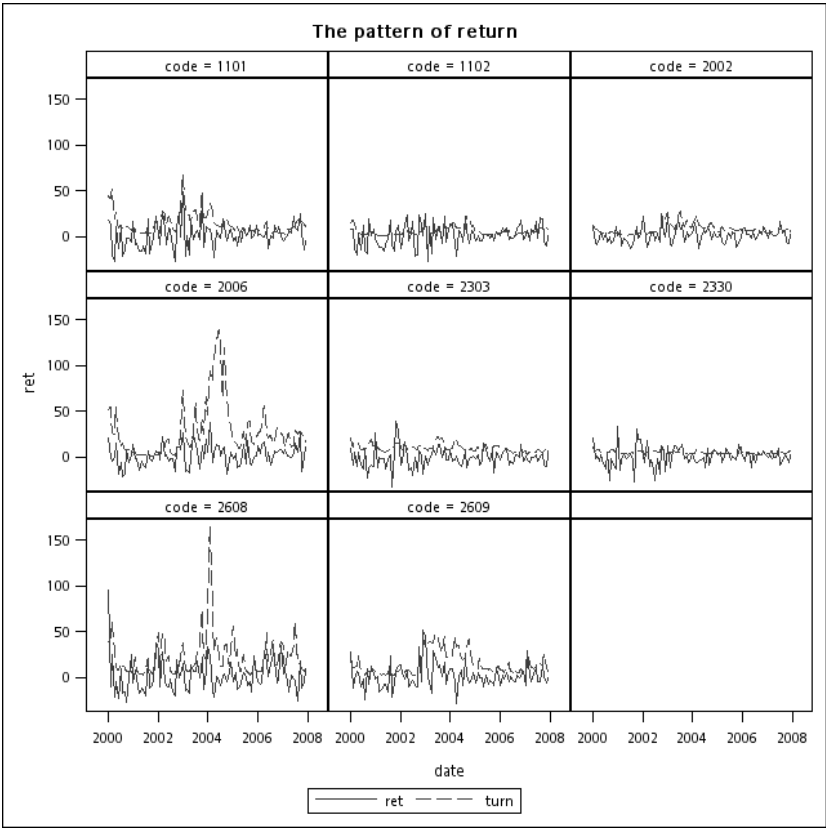


图 8-20

虽然 proc sgpanel 的绘图功能可以同时绘出几组股票的图形，但是其在画出不同组别的图时，却无

法将不同的股票完全塞在同一个绘图区中,要具有这样的功能,则可以使用如表 8-26 所示的 proc sgplot 的语法。

表 8-26

```
data a1;
set a;
if y=2001 then output;
run;
proc sgplot data=a1;
title "The pattern of return";
series x=date y=ret/ group=code;
run;
```

由于要一次绘制八只股票的报酬率趋势图,使用太长的期间会导致图形纠结在一起,反而不容易辨识,因此在此处只画出一年的 proc sgplot 输出结果,如图 8-21 所示。

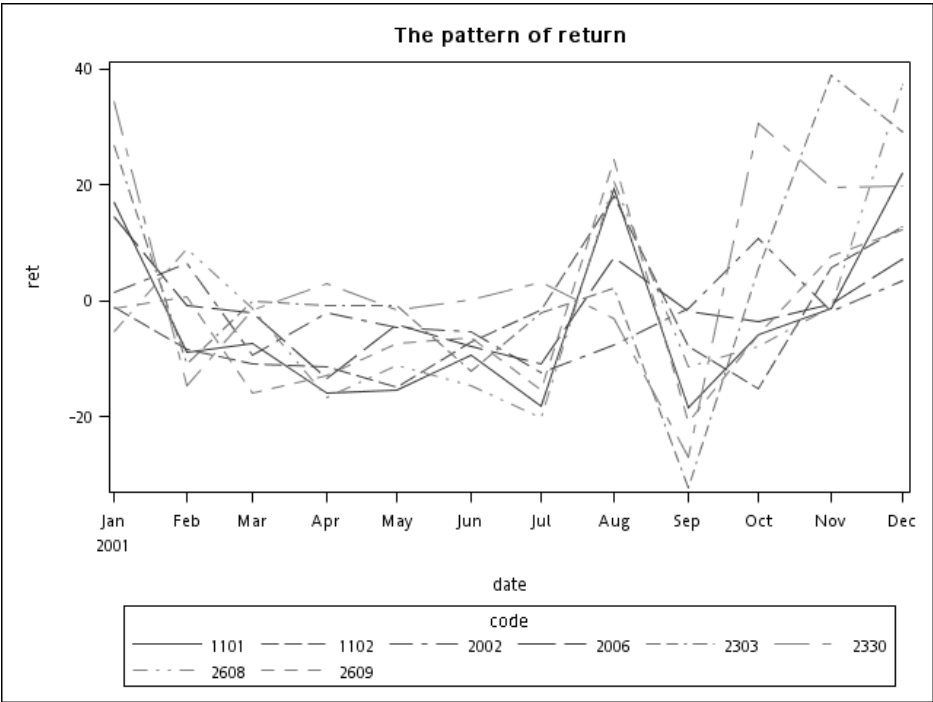


图 8-21

最后,介绍 proc sgpanel 功能最佳的绘图方式,由于 sgpanel 是用几个不同的 Panel 来绘图并且聚集在一起的,因此使用 sgpanel 的 panelby 的功能,可以绘出漂亮的 conditional 的图形,在进行图形比较上,给用户提供相当大的帮助,以下采用 2 乘 2 的绘图方式,如表 8-27 所示。

表 8-27

<pre>proc means data=a noprint; var mv turn; by code ; output out=b(drop=_type_ _freq_) mean=size turnover; run; proc rank data=b out=b groups=2; var size; ranks sizerank; run; proc sort data=b;by sizerank; run; proc rank data=b out=b groups=2; var turnover; ranks turnrank; by sizerank; run; proc sort data=b;by code; run; proc sort data=a;by code; run; data a; merge a b;by code; run;</pre>
--

此处依照前面的章节的介绍，算出每只股票的平均规模和平均周转率，再依照规模和周转率分组，接着再将数据依照股票代码合并，就可得到每只股票所属的规模和周转率的组别，sgpanel 趋势给图的语句如表 8-28 所示。

表 8-28

<pre>proc sgpanel data=a; panelby sizerank turnrank ; series x=date y=ret/group=code; run;</pre>
--

最后，再将数据依照组别分组绘图，由于共有八只股票，因此每一组里面会有两只股票，藉此可以做一点分析比较，sgpanel 分组趋势图的结果如图 8-22 所示。

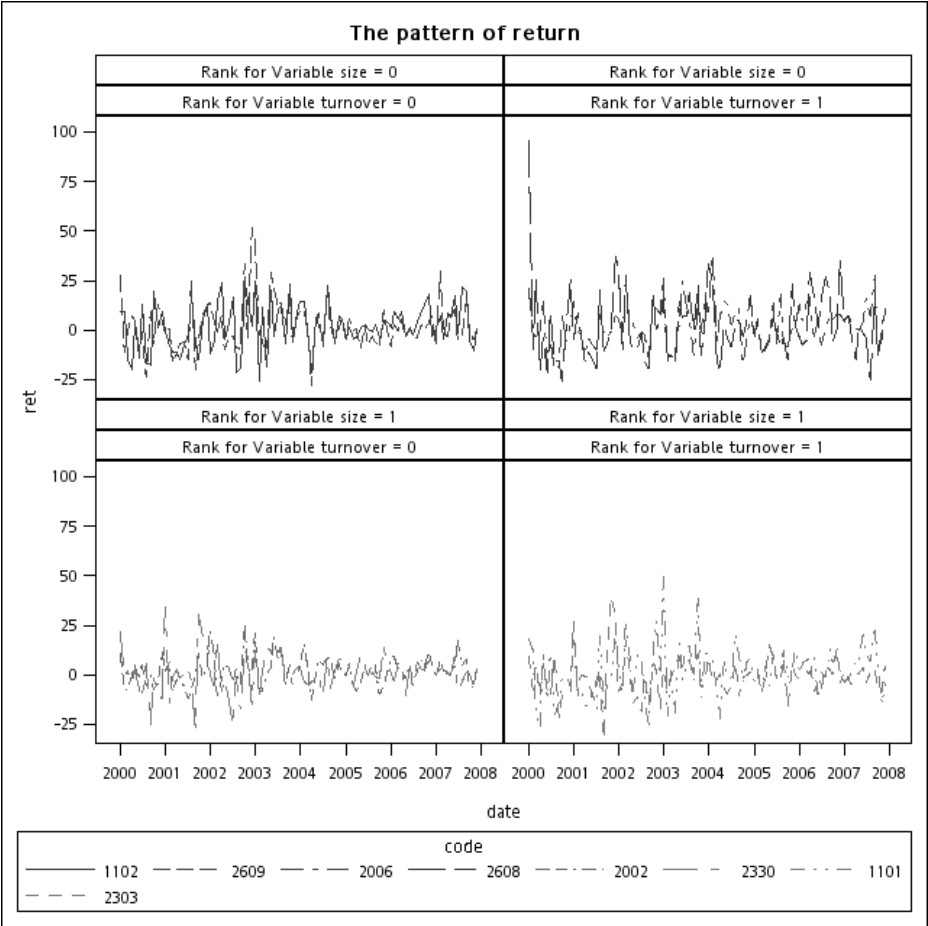


图 8-22

由图 8-22 来看，确实是有比较的功能，但是 SAS 将 Label 输出到结果中，然而使用原始变量名也会有问题，读者依然不了解 sizerank=0 或者 turnrank=1 代表什么意思。而 SAS 提供了 proc format 这个程序，可以将 0 与 1 进行编码，proc format 语句的介绍如表 8-29 所示。

表 8-29

<pre>proc format; value size 0="Small firm" 1="Large Firm"; value turn 0="Low Turnover" 1="High Turnover"; run;</pre>

在此处声明了两种格式，一种格式命名为 size，当数据为 0 时，将其称为小公司，数据为 1 时，将其称为大公司；另一种格式为 turn，0 为低周转率，1 为高周转率，声明这两种格式有何意义呢？如果读者仍有印象，可以思考一下日期的时间格式，日期的时间格式也是相同的概念，当值为 0 时，只要声明时间的格式语法，就可以知道 0 表示为 1960 年 1 月 1 日，因此这两种格式可以用如表 8-30 所示的 format 语句的语法来执行，format 执行的结果如图 8-23 所示。

表 8-30

```
data a;
set a;
format sizerank size.;
format turnrank turn.;
run;
```

date	size	turnover	sizerank	turnrank
00JAN	55310.677083	14.298958333	Large Firm	High Turnover
00FEB	55310.677083	14.298958333	Large Firm	High Turnover
00MAR	55310.677083	14.298958333	Large Firm	High Turnover
00APR	55310.677083	14.298958333	Large Firm	High Turnover
00MAY	55310.677083	14.298958333	Large Firm	High Turnover
00JUN	55310.677083	14.298958333	Large Firm	High Turnover
00JUL	55310.677083	14.298958333	Large Firm	High Turnover

图 8-23

这样，分组的数据立刻变为小公司与大公司，在使用上 proc format 类似于宏语法的 %mend name; %mend; 的写法，亦即使用者先撰写一个格式名称，接着在 data set 中使用 format 这个语法呼叫格式，使用时是 format 变量名格式名.，要特别注意的是格式名后紧接着的小数点，但是这部分并不需要担心，因为在程序编写中，如果编写正确，格式名将 是淡绿色的字样。然后，采用如表 8-31 所示的语句进行分组绩效绘图。

表 8-31

```
proc sgpanel data=a;
panelby sizerank turnrank ;
series x=date y=ret/group=code;
run;
```

接着，可以画出如图 8-24 所示的 format 后续效绘图结果的图样，虽然可以看出小公司、大公司以及高周转率、低周转率的组别，但是却仍有部分不便，也就是说 SAS 的 label 变量，即使是取消 label，仍需要解释 sizerank 以及 turnrank 的意义，反过来思考，用户是否仍可利用 SAS 的 label 呢？

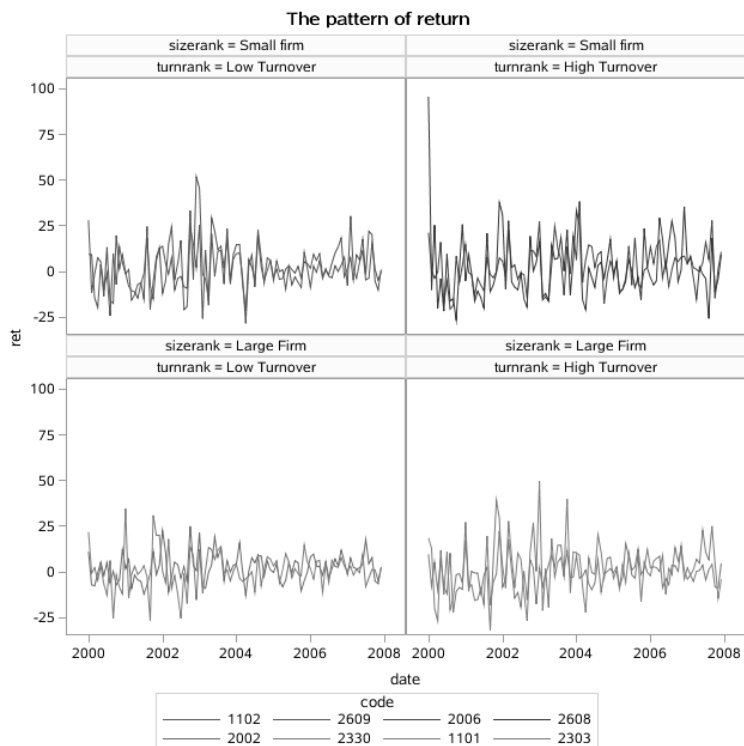


图 8-24

虽然在前面的章节，笔者强烈建议读者在撰写语法时要将 SAS 的 label 取消掉，但是在进行表格输出以及绘图的时候，反而需要利用 label 的语法，如此可以将结果输出得更漂亮¹，因此在绘图时，反而需要将 label 开启。变量栏位的 label 执行语句如表 8-32 所示。

表 8-32

options label;
data a;
set a;
label sizerank="Size";
label turnrank="Vlume";
label date="date";
label ret="Return (%)";
run;

1 当然，根据本书的做法，读者也可自行在 Excel 修改数据格式，但是如果使用 SAS 绘图，由于已经变成图片文件的形式，我们无法更改图片数据，因此必须先进行格式修正。

在此处，一次将四个变量重新做了 label，其做法与进行 proc format 类似，用户可以探讨 label 以后的 SAS 文档，以及其绘图结果。变量 label 执行结果如图 8-25 所示。执行 label 后的绘图结果如图 8-26 所示。

date	size	turnover	Size	Volume
00JAN	55310.677083	14.298958333	Large Firm	High Turnover
00FEB	55310.677083	14.298958333	Large Firm	High Turnover
00MAR	55310.677083	14.298958333	Large Firm	High Turnover
00APR	55310.677083	14.298958333	Large Firm	High Turnover
00MAY	55310.677083	14.298958333	Large Firm	High Turnover
00JUN	55310.677083	14.298958333	Large Firm	High Turnover
00JUL	55310.677083	14.298958333	Large Firm	High Turnover

图 8-25

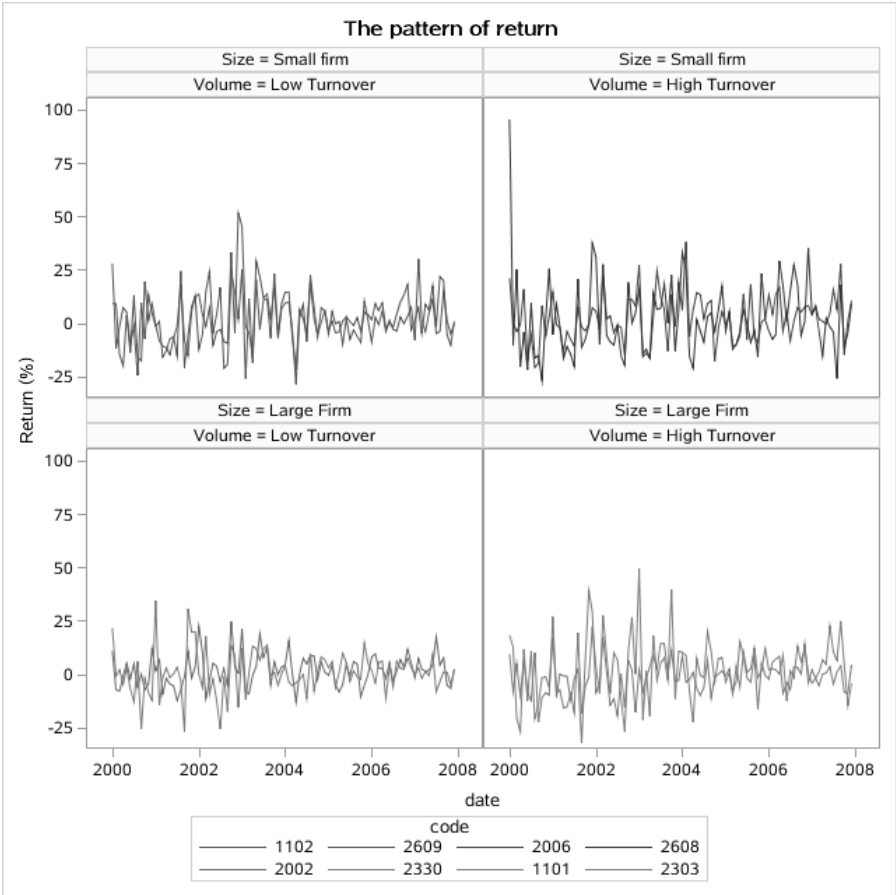


图 8-26

到此，几乎完成了所有的趋势图绘图格式介绍，最后，特别为学术格式的图形进行语法介绍，由于 SAS 为了报告方便起见，趋势图都会上色，但是在学术期刊上，由于打印出来的文档都是黑白的，使用颜色区别会产生问题，因此 SAS 提供了学术格式图形，会使得趋势图都变为灰阶。采用 journal 绘图格式语句如表 8-33 所示。

表 8-33

<pre>ods listing style=journal; proc sgpanel data=a; panelby sizerank turnrank ; series x=date y=ret/group=code; run;</pre>

只要执行 `ods listing style=journal;`，就可以使输出的图形符合期刊论文所用，但是在制作 PPT 上，则不一定需要使用该语法。最后提醒读者，虽然 SAS 提供了 `proc sg` 群组的语法¹，为了报告方便，将像素设定为 640×640 的像素，在存储文档时只需要几 KB 的容量即可，但是牺牲了设定存盘位置以及文件名语法，如果我们重复绘图，会导致开启最新的程序的所在位置的逻辑库一直产生图形文档，而由于名称无法自由命名，使用者必须要随时了解跑出来的图形文档的用途。

8.5 离群值的处理 (winsorize)

在实证分析中，常会需要删除离群值，而在财务研究中，除了删除离群值外，甚至会将高于 99 百分位数的值以 99 百分位数取代，低于 1 百分位数的值以 1 百分位数取代，这一类型的相关技术，通常称之为离群值的处理 (winsorize)。以下以 `example_8_5.sas` 为例来介绍这两种处理离群值的方法。离群值整理语句如表 8-34 所示。

表 8-34

<pre>libname aa "D:\The Application of SAS in Financial Research\SAS data\monthly price"; data a; set aa.price; run; proc univariate data=a noprint; var mv; output out=winsor pctlpts=1 99 pctlpre=limit_ pctlname=low high; /*pctlpts 要求输出该变量的第几百分位数的数值 pctlpre 输出的数值的变量名要求前导名称</pre>

1 SAS9.2 版新增了四个进阶版本的绘图语法。

续表

<pre>pctlname 输出的数值的命名 由于要求有前导名称所以命名就变为 limit_low limit_high */ run; data a_delete; set a; if _n_=1 then set winsor; /*这是一个有趣的语法 当观测值为 1 时 读取 winsor 的数据 所以 SAS 就会在读完 a 这个文档后 重新再读取 winsor 的数据 并且反复读完 */ if limit_low>mv>limit_high then delete; drop limit_high limit_low; run; data a_replace; set a; if _n_=1 then set winsor; if mv<limit_low then mv=limit_low; if mv>limit_high then mv=limit_high; drop limit_high limit_low; run; /* 这样 同时就完成删除以及取代的 winsorize 语法 */</pre>
--

虽然这样子就完成了 winsorize 的语法，但在使用上都需要反复修改，会有所不便，所以先思考有哪些变量是需要考虑的，第一，要进行 winsorize 的变量，可设一个宏变量 var 取代之，第二是来源文档，这可以采用 file 来取代之，第三与第四分别是变量数值的最低与最高百分位数，此处采用 low 以及 high 来修改。因此可以将 winsorize 改写为方便使用的宏语法，全样本离群值整理的宏语句如表 8-35 所示。

表 8-35

<pre>%macro winsorize(file,var,low,high); proc univariate data=&file noprint; var &var; output out=winsor pctlpts=&low &high pctlpre=limit_ pctlname=low high; run;</pre>

```

data &file._delete;
/*
注意 此处是&file._
而非      &file_
*/
set &file;
if _n_=1 then set winsor;

if limit_low>&var>limit_high then delete;
drop limit_high limit_low;
run;
data &file._replace;
set &file;
if _n_=1 then set winsor;
if &var<limit_low then &var=limit_low;
if &var>limit_high then &var=limit_high;
drop limit_high limit_low;
run;
proc datasets noprint;
delete winsor;
quit;
/*
winsor 已经使用完毕 所以删除之避免占据硬盘空间
*/
%mend;
%winsorize(a,mv,2.5,97.5);
%winsorize(a,ret,0.5,99.5);
/*使用上 最好是对称的*/

```

上述语法乍看之下似乎是合理的，但是深究一下，财务数据还有一个重要的时间变量，若 99 百分位的数值发生在 2012 年 4 月，而 2010 年的数值都大于 99 百分位数，这样删除或取代恐怕不太合理，因此上述语法比较适合在单期，或者单一公司的序列使用，若是面板数据的类型则需要纳入时间变量作为考虑，因此将 winsorize 的语法再度改写。依照组别（时间）的 winsorize 宏如表 8-36 所示。

表 8-36

```

%macro winsorize_time(file,delete,var,time,low,high);

  proc sort data=&file;by &time;
  run;

  proc univariate data=&file noprint;
    var &var;
    by &time;
    output out=winsor pctlpts=&low &high pctlpre=limit_
      pctlname=low high;
  run;

  /*revise 20150728*/
  %if &delete=1 %then %do;
    data &file;
      merge &file winsor;by &time;
      if limit_low<&var<limit_high then delete;
      drop limit_high limit_low;
    run;
  %end;
  %else %do;
    data &file;
      merge &file winsor;by &time;
      if &var<limit_low then &var=limit_low;
      if &var>limit_high then &var=limit_high;
      drop limit_high limit_low;
    run;
  %end;

  /*revise 20150728*/
  proc datasets noprint;
    delete winsor;
  quit;

%mend;

%winsorize_time(a,0,mv,y m, 2.5,97.5);
%winsorize_time(a,0,ret,y m,0.5,99.5);
/*使用上 最好是对称的

  特别注意 &file._delete 的文件 其数据已经被删除掉
  所以变量的删除顺序不同 会影响最终文档的结果
*/

```

采用上述语法，就可以将变量进行横截面上数值的 winsorize，而这个方法也比较符合财务研究的性质，本章将该宏语法保存为独立文件，若读者需要可采用%include 的来读取，而使用单一公司的跨期数据，或单期的横截面研究，则可自行复制前一个宏语法，自行另存为可呼叫的 winsorize 宏程序集，就不用再另行撰写相关的语法。

winsorize 宏语法使用介绍如表 8-37 所示。

表 8-37

<pre>%inc "D:\The Application of SAS in Financial Research\CH08\program_cn\winsorize.sas"; %winsorize(a,0,ret, 2.5,97.5); %winsorize_time(a,0,turn,y m, 2.5,97.5);</pre>
--

在该语法中，%winsorize 是提供给单一期间的数据使用的，%winsorize_time 是提供给多期数据使用的，a 表示文档，0 表示使用取代的句法，接下来则是变量以及数值的百分位数，读者可以自己研究并使用。

8.6 总结

本章节介绍了 proc means、proc univariate、proc corr 这三个描述统计的程序，并且提供了一个简单好用的相关系数以及 winsorize 宏程序，可以帮助读者在进行数据分析时，快速地求得需要的数据。除此之外，本章还介绍了趋势图的绘图语法，当然绘图语法并非只有趋势图，仍有其他绘图的图形可以使用，有兴趣的读者可以持续专研。

事实上，描述统计的程序是相当常用的语法，因为描述统计的语法提供了几个推论统计的统计量，在结合其他语法时，还可以画出有效边界的线段，用户除了了解程序计算出来的结果之外，亦可以思考这些结果的其他应用的情况。

第 9 章

两群体差异性检定

在财务研究中通常会进行次样本（sub-sample）的研究，会将样本区分为不同类型的公司来分析，此时会初步探讨这些公司是否会有差异，并进行两群体的差异性检定，本章介绍如何撰写这方面的语法。

9.1 均值检定

由于财务数据有跨期间的特性，除了简单的分组比较外，还有不同年度的分组比较，以下采用 example_9_1.sas 来介绍该语法。数据分组语句如表 9-1 所示。

表 9-1

```
libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price';  
data a;  
set aa.price;  
if m=12 and y>2000 then output;  
run;  
proc sort data=a;by y;  
run;  
proc rank data=a out=a groups=2;  
var mv;  
ranks size;  
by y;  
run;
```

为了避免数据过于繁多，此处仅为用户 2000 年以后的数据，并且将公司每年依照市值分成两组。将数据进行 student t 检定的语句如表 9-2 所示。

表 9-2

```
proc ttest data=a;  
var volum value ret turn pb;  
class size;  
ods output Statistics=t;  
run;
```

这部分的语法直接将数据的各个变量分成两组规模不同的公司进行比较，SAS 执行后的结果，如图 9-1 所示。

	Variable	Class	N	Mean	LowerCLMean	UpperCLMean	StdDev	LowerCLStdDev	UpperCLStdDev	UMF01LowerCLStdDev	UMF01UpperCLStdDev	StdErr
1	volum	0	2583	24.7123	23.0416	26.3831	43.3036	42.1542	44.5180	42.1488	44.5121	0.8530
2	volum	1	2586	169.1	158.9	179.2	263.9	256.9	271.3	256.9	271.2	5.1892
3	volum	Diff (1-2)	-	-144.3	-154.7	-134.0	189.1	185.6	192.9	185.6	192.9	5.2616
4	value	0	2583	313.2	291.2	335.1	569.4	554.3	585.4	554.2	585.3	11.2037
5	value	1	2586	5207.5	4796.1	5610.9	10668.9	10385.9	10967.9	10384.6	10966.5	209.8
6	value	Diff (1-2)	-	-4894.4	-5306.5	-4482.2	7557.0	7414.1	7705.6	7413.6	7705.1	210.2
7	ret	0	2581	6.8698	6.0240	7.7156	21.9135	21.3316	22.5282	21.3288	22.5253	0.4313
8	ret	1	2586	7.3324	6.7163	7.9485	15.9777	15.5338	16.4255	15.5518	16.4233	0.3142
9	ret	Diff (1-2)	-	-0.4626	-1.5084	0.5833	19.1738	18.8111	19.5508	18.8099	19.5496	0.5335
10	turn	0	2583	15.1437	14.3052	15.9823	21.7349	21.1580	22.3444	21.1553	22.3415	0.4277
11	turn	1	2586	24.3994	23.2771	25.5216	29.1036	28.3315	29.9193	28.3279	29.9153	0.5723
12	turn	Diff (1-2)	-	-9.2556	-10.6565	-7.8548	25.6870	25.2012	26.1920	25.1996	26.1903	0.7146
13	PB	0	2523	1.1388	1.1020	1.1757	0.9432	0.9179	0.9700	0.9178	0.9699	0.0188
14	PB	1	2582	2.0167	1.9564	2.0770	1.5628	1.5213	1.6066	1.5211	1.6064	0.0308
15	PB	Diff (1-2)	-	-0.8778	-0.9489	-0.8068	1.2942	1.2696	1.3198	1.2695	1.3197	0.0362

图 9-1

由数据上，用户仅需要变量（ variable ）、分组变量（ class ）、平均值（ mean ）以及标准误（ stderr ）即可完成均值差异性检定表格，因此后续语法利用这些变量，便可以完成接下来的程序。将 t 进行统计分析整理的语句如表 9-3 所示。

表 9-3

```
data t;
  set t;
  order=int((_n_-1)/3);
  tv="("||left(round(mean/stderr,0.01)||")");
  if abs(mean/stderr)>2.58 then m=round(mean,0.001)||"***";
  else if abs(mean/stderr)>1.96 then m=round(mean,0.001)||"***";
  else if abs(mean/stderr)>1.65 then m=round(mean,0.001)||"*";
  else m=round(mean,0.001)||"";
  if mod(_n_,3)=1 then class='G1';
  if mod(_n_,3)=2 then class='G2';
  if mod(_n_,3)=0 then class='Dif';
run;
```

在这部分语法中，order 是为了在转置的时候，仍然能够使 SAS 依照我们想要的顺序排列，而在 ttest 的程序中，各个变量是以 3 个一组为序列的，因此只要将观测值减去 1 以后再除以 3，就可以顺利进行了。统计表格转置处理语句如表 9-4 所示。

表 9-4

```
proc transpose data=t out=t(drop=order _name_);
  var m tv;
  id class;
  by order variable;
run;
data t;
  set t;
  if mod(_n_,2)=0 then variable="";
run;
```

接着便是进行转置以及细节的修正，先探讨表格最后呈现的样子，如图 9-2 所示。
图 9-2 便是程序执行完毕的结果，读者仅需要输出到 Excel 以后，再将 _0_、_1 以及 Diff__1_2_ 这 3 个变量作适当的命名，就可以直接使用。接下来介绍如何分年度进行检定。分年度进行 t 检定的语句如表 9-5 所示。

	Variable	G1	G2	Dif
1	volum	24.712***	169.052***	-144.339***
2		(29)	(32.58)	(-27.43)
3	value	313.174***	5207.537***	-4894.363***
4		(27.95)	(24.82)	(-23.28)
5	ret	6.87***	7.332***	-0.463
6		(15.93)	(23.34)	(-0.87)
7	turn	15.144***	24.399***	-9.256***
8		(35.41)	(42.63)	(-12.95)
9	PB	1.139***	2.017***	-0.878***
10		(60.65)	(65.57)	(-24.23)

图 9-2

表 9-5

```
proc sort data=a;by y size;
run;
proc ttest data=a;
var volum value ret turn pb;
class size;
by y;
ods output Statistics=t;
run;
```

由于要分年度检定，所以一定要先依照年份进行排序，并且在进行 ttest 时，多增加 by 这个语法，读者会怀疑，为何需要在此处依照 y、size 的顺序排序，在 ttest 时 size 并不会影响程序的进行，但这里是一个撰写宏语法的小技巧，是为了撰写分年度或不分年度皆可使用的 ttest 的宏，此处利用分组的变量，来帮助当日后不指定年度变量时，亦可让程序顺利进行，因为两群体检定一定要指定组别，所以依照其进行排序并不会特别干扰分析的结果。分年度 t 统计量执行结果的语句如表 9-6 所示。

表 9-6

```
data t;
set t;
order=int((_n_-1)/3);
tv="("||left(round(mean/stderr,0.01))||")";
if abs(mean/stderr)>2.58 then m=round(mean,0.001)||"***";
else if abs(mean/stderr)>1.96 then m=round(mean,0.001)||"**";
else if abs(mean/stderr)>1.65 then m=round(mean,0.001)||"*";
else m=round(mean,0.001)||";";
if mod(_n_,3)=1 then class='G1';
if mod(_n_,3)=2 then class='G2';
if mod(_n_,3)=0 then class='Dif';
run;
```

此部分的语法并不会做任何的修正，所以继续修正下面的语法。依照年份进行转置疏理的语句如表 9-7 所示。

表 9-7

<pre>proc transpose data=t out=t(drop=order_name_); var m tv; id class; by y order variable; run; data t; set t; if mod(_n_,2)=0 then variable=""; run;</pre>

在这部分的语法中，只要把 by order variable 改成 by y order variable 就可以了，依照这样的逻辑概念，用户可以简单地撰写 ttest 的宏语法，这个宏语法需要考虑的是检定的变量串、小数点位数、分组变量、逐年变量、检定的来源文档以及输出文档，所以可以开始改写这些语法。进行 t 检定的宏语法变量准备如表 9-8 所示。

表 9-8

<pre>%macro ttest(variable,bit,class, year,infile,outfile); 程序语法 %mend;</pre>

第一步先这样撰写，然后逐步探讨要修改的程序语法。t 检定宏语法修改一如表 9-9 所示。

表 9-9

<pre>proc sort data=&infile; by &year &class; run; proc ttest data=&infile; var &variable; class &class; by &year; ods output Statistics=t; run;</pre>
--

此处由于 a 是来源文档，所以将其改成&infile，接着再将 y size 改成&year &class，再持续向下检验，将变量串、来源文档、分组变量以及年度变量都修改掉。t 检定宏语法修改二如表 9-10 所示。

表 9-10

```
data t;
set t;
order=int((_n_-1)/3);
tv="("||left(round(mean/stderr,0.01))||");
if abs(mean/stderr)>2.58 then m=round(mean,10**(-1*&bit))||'***';
else if abs(mean/stderr)>1.96 then m=round(mean,10**(-1*&bit))||'**';
else if abs(mean/stderr)>1.65 then m=round(mean,10**(-1*&bit))||'*';
else m=round(mean,10**(-1*&bit))||";
if mod(_n_,3)=1 then class='G1';
if mod(_n_,3)=2 then class='G2';
if mod(_n_,3)=0 then class='Dif';
run;
```

这部分的语法几乎完全不需要变更，在本书后续章节会重复出现类似的语法，唯一有可能要变更的就是要修改报告出来的小数点位数。t 检定宏语法更改三如表 9-11 所示。

表 9-11

```
proc transpose data=t out=t(drop=order _name_);
var m tv;
id class;
by &year order variable;
run;
data &outfile;
set t;
if mod(_n_,2)=0 then variable="";
run;
proc datasets noprint;
save &infile &outfile;
quit;
```

在此处，修正了&year 以及&outfile，并且要求最终结果只保留&infile 以及&outfile，到此就将整个宏改写完成了。

介绍检定宏语法的使用，如表 9-12 所示。

表 9-12

```
%ttest(volum value ret turn pb,4,size,y,a,b);
%ttest(volum value ret turn pb,2,size, ,a,c);
```


至此，读者就可以轻松地完成两群体比较的表格了，但如果有特殊需求，便需要自己先将数据进行整理才能够进行比较。

9.2 中位数检定

在前一节介绍了 t 检定的宏语法，本节以 example_9_2.sas 为例来介绍中位数检定宏语法的撰写。中位数检定的语法如表 9-13 所示。

表 9-13

<pre>option nolabel; libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price'; data a; set aa.price; if m=12 and y>2000 then output; run; proc sort data=a;by y; run; proc rank data=a out=a groups=2; var mv; ranks size; by y; run; proc npar1way data=a; var volum value ret turn pb; class size; ods output WilcoxonTest=t; run;</pre>
--

在 SAS 中，中位数检定的语法仅报告两群体是否有显著差异，并没有其中位数相关的数值，所以必须使用其他的语法补充，前面介绍了 proc univariate 这个程序，它具有中位数相关的统计量，为了得到两组的中位数以及中位数差异，接下来需要使用 proc univariate 语法的中位数检定结果进行整理，如表 9-14 所示。

表 9-14

<pre>proc univariate data=a noprint; var volum value ret turn pb; class size; output out=m median=volum value ret turn pb; run;</pre>

```

proc transpose data=m out=m;
var volum value ret turn pb;
id size;
run;
data t;
set t;
retain order 0;
if variable^=lag(variable) then order=order+1;
run;
proc transpose data=t out=t;
var nvalue1;
id name1;
by order variable;
run;
data m;
merge t m;
if pl_wil<0.01 then dif=round(_0_1,0.001)||'***';
else if pl_wil<0.05 then dif=round(_0_1,0.001)||'**';
else if pl_wil<0.1 then dif=round(_0_1,0.001)||'*';
else dif=round(_0_1,0.001)||'';
rename _0=G1;
rename _1=G2;
keep variable _0 _1 dif;
run;

```

如此便可以得到中位数检定执行的结果，如图 9-3 所示。

Variable	G1	G2	dif
volum	9	80	-71***
value	101	1919.5	-1818.5***
ret	3.11	4.29	-1.18***
turn	6.96	13.71	-6.75***
PB	0.92	1.59	-0.67***

图 9-3

在中位数检定中，本书使用的是 wilcox 的检定值，但由于跟常见的检定值不同，在表格中仅在差异性检定上标示是否有显著的星号，接下来介绍分年度进行中位数检定，如表 9-15 所示，然后介绍如何使用宏。

表 9-15

<pre>proc sort data=a;by y size; run; proc npar1way data=a; var volum value ret turn pb; class size; by y; ods output WilcoxonTest=t ; run;</pre>

在该部分，跟 t 检定一样，只改写了 y 的部分，其理由仍是日后可以为顺利改写的宏语法所用。分年度中位数检定结果整理的语法如表 9-16 所示。

表 9-16

<pre>proc univariate data=a noprint; var volum value ret turn pb; class size; by y; output out=m median=volum value ret turn pb; run; proc transpose data=m out=m; var volum value ret turn pb; id size; by y; run; data t; set t; retain order 0; if variable^=lag(variable) then order=order+1; run; proc transpose data=t out=t; var nvalue1; id name1; by y order variable; run; data m; merge t m; if pl_wil<0.01 then dif=round(_0-_1,0.001) ***; else if pl_wil<0.05 then dif=round(_0-_1,0.001) **;</pre>

续表

```
else if pl_wil<0.1 then dif=round(_0-_1,0.001)||'*';  
else dif=round(_0-_1,0.001)||";  
rename _0=G1;  
rename _1=G2;  
keep y variable _0 _1 dif;  
run;
```

此部分的语法，依然是将语法加上 by y 便可以解决了，接下来为读者介绍如何使用本书改写完成的中位数检定¹方法。分年度中位数检定宏语法的使用如表 9-17 所示。

表 9-17

```
%npar(volum value ret turn pb,4,size,y,a,b);  
%npar(volum value ret turn pb,2,size, ,a,c);
```

采用该宏，便可以快速地进行中位数检定。

9.3 两群体检定宏进阶用法

在前两节中，仅使用分年度的比较，实际上分年度的概念也适用于分组的情况，在本节将样本持续分类，以了解本宏语法不仅是只能用在逐年分析。均值与中位数检定宏语法的使用如表 9-18 所示。均值检定的结果如图 9-4 所示。

表 9-18

```
libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price';  
data a;  
set aa.price;  
if m=12 and y>2000 then output;  
run;  
proc sort data=a;by y;  
run;  
proc rank data=a out=a groups=2;  
var mv turn;  
ranks size turnr;  
by y;  
run;
```

¹ 有兴趣改写的读者，可先自行改写宏，再自行检验是否可以使用，改写方法如同前一节的方法。

续表

```
%inc "D:\The Application of SAS in Financial Research\CH09\program_cn\ttest.sas";
%inc "D:\The Application of SAS in Financial Research\CH09\program_cn\mpar.sas";
%ttest(volum value ret turn pb,4,size,y turnr,a,b);
%ttest(volum value ret turn pb,2,size, turnr,a,c);

proc export data=b outfile="D:\The Application of SAS in Financial Research\CH09\output\parameter_test"
dbms=xlsw
replace;
sheet='b';
quit;

proc export data=c outfile="D:\The Application of SAS in Financial Research\CH09\output\parameter_test"
dbms=xlsw
replace;
sheet='c';
quit;

%mpar(volum value ret turn pb,4,size,y pbr,a,b);
%mpar(volum value ret turn pb,2,size, pbr,a,c);

proc export data=b outfile="D:\The Application of SAS in Financial Research\CH09\output\mpar_test"
dbms=xlsw
replace;
sheet='b';
quit;

proc export data=c outfile="D:\The Application of SAS in Financial Research\CH09\output\mpar_test"
dbms=xlsw
replace;
sheet='c';
quit;
```

	turnr	Variable	_0	_1	Diff _1_2
1	0	volum	7.34***	92.31***	-84.98***
2	0		(19.06)	(17)	(-18.98)
3	0	value	63.69***	3000.36***	-2936.67***
4	0		(26.7)	(11.98)	(-14.3)
5	0	ret	5.08***	3.87***	1.21
6	0		(8.79)	(11.33)	(1.6)
7	0	turn	3.81***	5.55***	-1.74***
8	0		(40.22)	(48.66)	(-11.69)
9	0	PB	1.08***	1.86***	-0.78***
10	0		(38.82)	(42.35)	(-15.81)
11	1	volum	50.58***	220.59***	-170.01***
12	1		(28.77)	(29.04)	(-18.12)
13	1	value	684.51***	6689.93***	-6005.41***
14	1		(29.38)	(22.15)	(-16.27)
15	1	ret	9.54***	9.66***	-0.12
16	1		(15.08)	(20.85)	(-0.16)
17	1	turn	32.01***	37.06***	-5.05***
18	1		(39.57)	(45.91)	(-4.25)
19	1	PB	1.23***	2.12***	-0.89***

图 9-4

由该部分语法显示，读者可以将其用于不同交易量下，不同规模组合的各个变量的检定，接下来探讨逐年逐交易量的表格。分年度均值检定的结果如图 9-5 所示。

	y	turn	Variable	_0	_1	Diff _1_2_
1	2001	0	volum	13.6091***	109.0174***	-95.4083***
2	2001	0		(8.43)	(4.98)	(-5.98)
3	2001	0	value	49.9727***	4099.3565***	-4049.3838***
4	2001	0		(11.42)	(3)	(-4.1)
5	2001	0	ret	21.1404***	16.5643***	4.5761*
6	2001	0		(12.35)	(9.15)	(1.69)
7	2001	0	turn	4.7088***	6.272***	-1.5632***
8	2001	0		(16.41)	(15.47)	(-3.17)
9	2001	0	PB	0.7032***	1.325***	-0.6218***
10	2001	0		(12.92)	(9.6)	(-4.97)
11	2001	1	volum	59.5739***	339.7091***	-280.1352***
12	2001	1		(13.1)	(11.27)	(-6.69)
13	2001	1	value	784.4087***	10897.7545***	-10113.3458***
14	2001	1		(11.65)	(9.69)	(-6.49)
15	2001	1	ret	33.4263***	32.7064***	0.7199
16	2001	1		(14.16)	(20.1)	(0.25)
17	2001	1	turn	49.62***	66.903***	-17.283***
18	2001	1		(15.65)	(21.26)	(-3.51)
19	2001	1	PB	1.1997***	2.4346***	-1.2349***
20	2001	1		(16.11)	(17.11)	(-6.03)

图 9-5

这部分的结果说明逐年逐交易量的检定可以很顺利进行，而且可以相当灵活地使用该宏，有需要的读者可以多思考其使用的范围。

9.4 总结

本章介绍财务研究上最常使用的两群体检定的语法，虽然只是简单的检定，但是可以快速地呈现出研究表格，相信这对研究有很大的帮助。

第 10 章

投资组合与报酬率检定

本章主要介绍投资组合理论，首先介绍在一般财务教材中提到的多元化投资组合分散风险的极限，接着介绍如何绘制有效边界；并介绍在学术研究中最常接触到的投资组合的分组方式，希望通程序语法教学配合文献的阅读，让大家能够了解如何将文章所描述的流程转换成 SAS 程序，进而进行实证分析，而区分出投资组合以后，可提供一些回归模型所需要的变量，例如 Fama and French (1993) 的 3 因子模型以及 Carhart (1997) 的 4 因子模型是采用投资组合的报酬率来计算因素负荷量的。在本章的最后，由于投资组合是一时间序列的报酬率，可能会有自相关的现象，在学术上常使用 Newey and West 的方式来进行调整，因此也讨论了 Newey and West 格式化语法的问题。

10.1 投资组合股票的数目与风险

在投资组合理论中，最常提到的就是要进行多元化投资以分散风险，但是到底要用到多少只股票才能分散风险，以及风险到底可以降低到多少？可以降低 0，完全没有风险吗？或者是有其极限。

在投资组合中，风险常以股票的方差或者标准偏差来表示，要了解多元化投资分散风险的能力，就要先明白投资组合的方差该如何计算。首先，检验两项资产的情形。

若有两项资产 x_1 、 x_2 ，报酬率分别为 r_1 、 r_2 ，资产报酬率的方差分别为 σ_1^2 与 σ_2^2 ，今分别投资 w_1 、 w_2 的比例于资产 x_1 、 x_2 中，其投资组合的报酬率方差是多少？

$$\begin{aligned} \text{Var}(w_1r_1+w_2r_2) &= \text{Var}(w_1r_1) + \text{Var}(w_2r_2) + 2\text{Cov}(w_1r_1, w_2r_2) \\ &= w_1^2 \text{Var}(w_1r_1) + w_2^2 \text{Var}(w_2r_2) + 2w_1w_2\text{Cov}(r_1, r_2) \end{aligned}$$

若有 n 项资产，方差则为 $\sum_{i=1}^n w_i^2 \text{Var}(n) + \sum_{i=1}^n \sum_{j=1, j \neq i}^n w_j \text{Cov}(n, r_j)$ ，事实上，除了权重之外， $\sum_{i=1}^n w_i^2 \text{Var}(n) + \sum_{i=1}^n \sum_{j=1, j \neq i}^n w_j \text{Cov}(n, r_j)$ 为协方差矩阵的数据，为了看出资产数目增加对投资组合风险的影响，可以将各资产占总权重的比率皆设为 $1/n$ ，如此资产组合的总方差就等于 $\frac{1}{n^2} (\sum_{i=1}^n \text{Var}(n) + \sum_{i=1}^n \sum_{j=1, j \neq i}^n \text{Cov}(n, r_j))$ 。事实上在 `proc corr` 得到的协方差矩阵中，就包含了 $\frac{1}{n^2} (\sum_{i=1}^n \text{Var}(n) + \sum_{i=1}^n \sum_{j=1, j \neq i}^n \text{Cov}(n, r_j))$ 所有的数据，因此在计算投资组合报酬率的方差时，仅需要将协方差矩阵所有的数值加总即可，以下便介绍如何用 SAS 来进行多元化风险的计算。

本节介绍 `example_10_1.sas`，整个程序语法分为 3 个部分，第 1 部分是整理股票报酬率数据，第 2 部分是多元化宏程序，第 3 部分为绘图。股票报酬率数据整理的语句如表 10-1 所示。

表 10-1

```
option nonotes;
libname aa 'D:\The Application of SAS in Financial Research\CH10\data';
data a;
set aa.price;
if y=2006 then output;
keep y m code ret;
run;
proc sort data=a;by code y m;
run;
proc means noprint data=a;
var ret;by code;
output out=b(drop=_type_ _freq_) n=n;
```


续表

<pre>run; data a; merge a b;by code; if n=12 then output; run; data code; set a; if code=lag(code) then delete; d=1; keep code d; run;</pre>
--

在该部分中,使用 price 这个文档的数据,该数据包含了从 1980 年到 2007 年的股价报酬率等数据,为了程序讲解方便起见,仅要求输出 2006 年的数据,接着为了保证报酬率数据的完整性,利用了 proc means 程序中在计算均值时计算有效观测值的语法,整理过后便取得了最后进行的报酬率样本数据,以及 2006 年报酬率都存在的公司数,在 2006 年的上市公司中,报酬率整年存在的公司共有 701 家,为了在抽取样本后证明是研究所需的样本,此处设了变量 d=1,接下来,进行宏程序的介绍。分散投资组合宏语句如表 10-2 所示。

表 10-2

<pre>%macro diver; %do i=1 %to 200; %end; %mend diver;</pre>
--

首先,该宏命名为 diver,总共要执行 1 到 200 次,该次数的产生是针对投资组合的公司家数来计算的,亦即当 i=1 时,投资组合的公司家数只有 1 家,当 n=2 时,投资组合的公司家数有 2 家,根据随机原理,此处使用了随机抽样的程序来抽取样本。分散投资组合的随机抽样如表 10-3 所示。

表 10-3

<pre>proc surveyselect data=code out=b noprint method=srs n=&i seed=0 rep=1; run; data ret; merge a b;by code; if d=, then delete; run;</pre>

如要求 SAS 对 code 中进行简单随机抽样，种子设为 0，由系统开始时间作为种子，只抽样 1 次，由于 b 这个文档是由 code 抽取出来的，其文档有 d 这个变量，而 a 文档并没有 d 这个变量，所以该程序会删除掉非抽样公司的数据。分散投资组合的转置整理如表 10-4 所示。

表 10-4

<pre>proc sort data=ret;by y m code; run; proc transpose data=ret out=ret1; var ret; by y m; run;</pre>

此处是为了之后计算协方差矩阵所准备的，由于 proc corr 是针对变量进行相关系数的计算，在 ret 的数据中，变量只有 ret 而已，因此针对 ret 进行转置，并且依照每个月转置，报酬率就会变成 col1、col2、…的形式。均值方差整理的语句如表 10-5 所示。

表 10-5

<pre>proc corr data=ret1 cov outp=cov noprint; var col1-col&i; run; data cov; set cov; n=_n_; v=sum(of col1-col&i); retain var 0; var=var+v; std=((var)/n**2)**0.5; if _n_=&i then output; keep n std; run; proc append data=cov base=diver; run;</pre>

此处是计算总方差，并且开根号求其标准偏差，最后再将所有的数据重复地附加到 diver 这个 table 上，至此宏程序就算终结了。数据的绘图整理如表 10-6 所示。

表 10-6

<pre>goptions device=gif gsfname=gout xpixels=960 ypixels=720 ftext='Arial' htext=2 gunit=pct ctext=black csymbol=black; axis order=(0 to 200 by 20);</pre>

续表

```

filename gout "D:\The Application of SAS in Financial Research\CH10\output\portfolio.gif";
proc gplot data=diver;
symbol interpol=join value=dot;
plot std*n /
        hminor=1
        vminor=1
        lvref=1
        cvref=black
        caxis=black
        ctext=black;
run;
quit;

```

最后，针对计算出来的数据绘制趋势图，由于只进行一次投资组合抽取的测试，描出的趋势图并没有一般投资学或者财务管理的书籍那般平滑。投资组合的资产数目与风险如图 10-1 所示。

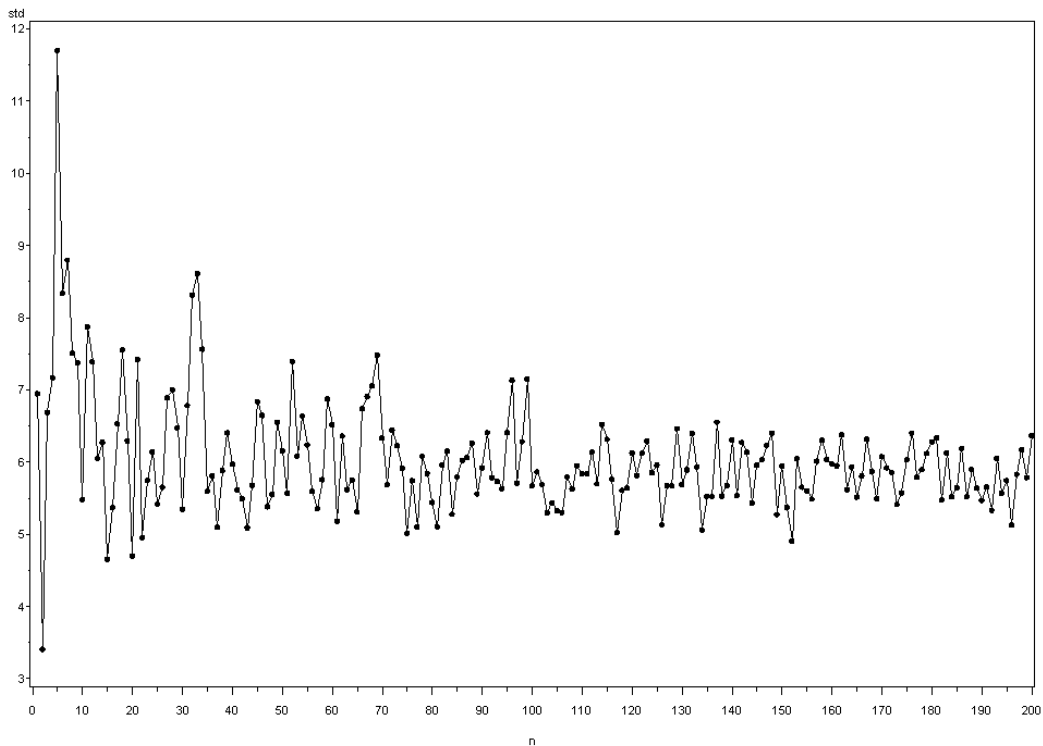


图 10-1

虽然风险的走势图并不平滑，但仍有一定的趋势，虽然看得出增加投资组合的证券数目可以降低风险，但是仍然有增加的现象，这是因为尝试次数不够多，而且此处只使用一年的期间来计算协方差，这可能由于 2006 年的整体市场报酬的连动性太高，使得使用这一年的数据会不够稳定。那么，可以多抽样几次，求其均值，以检验平均而言、由几家公司组成的投资组合的风险如何，可采用如表 10-7 所示的多次抽样建构投资组合的宏程序来完成。

表 10-7

<pre>%macro diversify(k); proc datasets noprint; delete diver; quit; %do l=1 %to &k; %diver; %end; proc sort data=diver;by n; run; proc means noprint data=diver; var std;by n; output out=diver mean=std; run; goptions device=gif gsfname=gout xpixels=960 ypixels=720 ftext='Arial' htext=2 gunit=pct ctext=black csymbol=black; filename gout "D:\The Application of SAS in Financial Research\CH10\output\portfolio&k..gif"; proc gplot data=diver; symbol interpol=join value=dot; plot std*n / hminor=1 vminor=1 lvref=1 cvref=black caxis=black ctext=black; run; quit;%mend diversify;</pre>	
---	--

该宏程序仅有宏 diver 要重复执行，这个例子也说明撰写的宏可以在内部执行另外一个宏。接着由于 diver 会有重复执行的投资组合数目及其风险，因此依照投资组合数目排序并计算其均值。最后再继续绘出图形，值得注意的是，输出文档的撰写，CH10\output\portfolio&k..gif，聪明的读者必然发现其中有 2 个小数点，虽然不明白为什么这么写，但是在笔者使用宏程序要输出图形文档时，若有使用了

指定变量作为文档名，则必须使用两个小数点。分散投资组合宏语法的使用如表 10-8 所示。

表 10-8

<code>%diversify(2);</code>
<code>%diversify(5);</code>
<code>%diversify(10);</code>
<code>%diversify(20);</code>
<code>%diversify(30);</code>
<code>%diversify(50);</code>
<code>%diversify(100);</code>

最后执行了该宏 2、5、10、20、30、50、100 次，就结果而言，重复抽取的次数越多，结果会越好，但是运行时间上也会越来越长，图 10-2 展示了执行 100 次分散投资组合建构的结果。

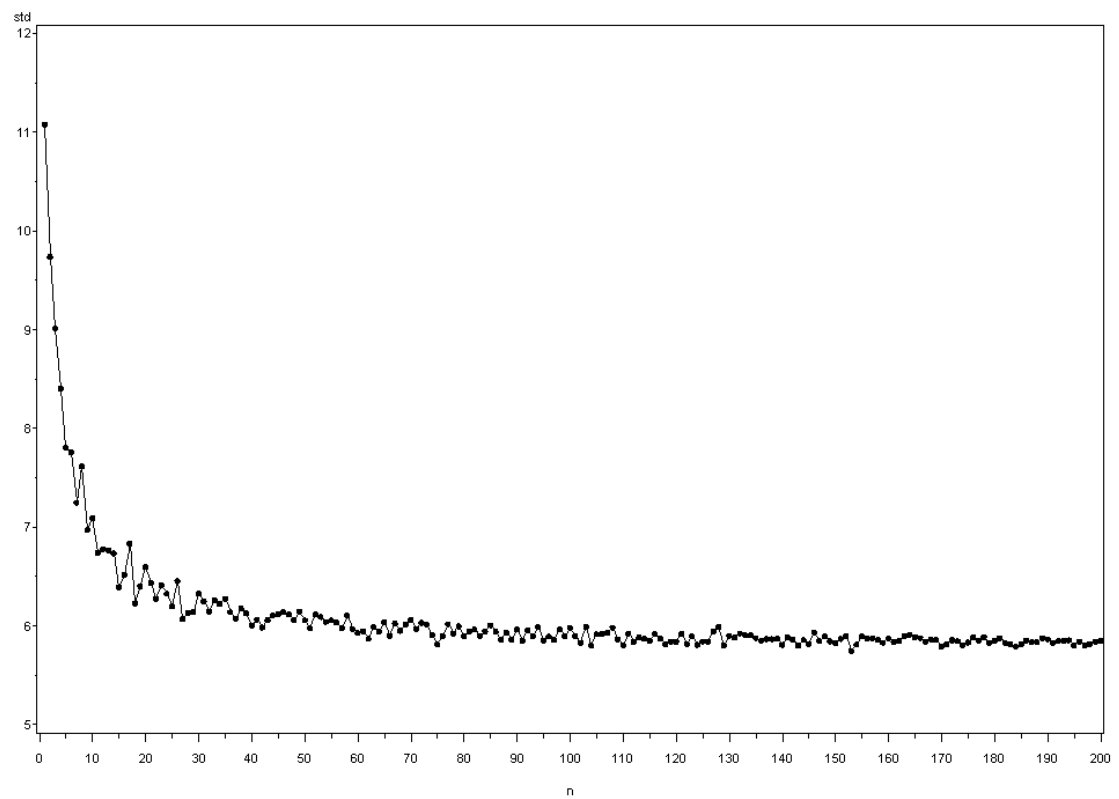


图 10-2

总之，抽取 100 次的平均结果发现，不管投资者如何增加证券数目，以月报酬率而言，投资组合的标准偏差约为 6，即这 6 就是不可分散的风险，也就是说投资组合为 15~20 只股票的话，风险就降

到 6 左右了，因此就台湾地区市场而言，要进行多元化投资以分散风险，大约只需要 15~20 只股票就足够了。

10.2 效率前缘的绘制

在了解了多元化投资组合以及分散风险的意义后，可以开始进行绘制有效边界，在一般财务管理以及投资学的教科书中，都会讲授由数个资产组合可以根据其权重搭配，最后画出有效边界。

然而，要画出有效边界却不是一件容易的事情，首先要取得多个资产，得到其预期报酬率、以及资产的协方差矩阵。接着开始一点一点地找出有效边界的点，例如先求得在报酬率 1% 下，这些资产组合在哪个权重搭配下的最低标准偏差，然后再求 1.1% 下的最低标准偏差，接着 1.2%……逐步地求出每一点。这些做法必须搭配撰写宏程序才可行，然而，要求得权重步骤的最佳做法是使用矩阵来计算。

若假设今有 n 项资产，有一个预期报酬率的矩阵 R ，以及协方差矩阵 Σ ，若投资组合 P 的各项资产权重矩阵为 ω ($N \times 1$)，其中 ω 的总和为 1， $\omega'R = R_p$ ，则经过推导之后，可求得

$$\omega = \Sigma^{-1}(R, I)((R, I)' \Sigma^{-1}(R, I))^{-1} \begin{pmatrix} R_p \\ 1 \end{pmatrix}$$

可以发现最重要的是协方差矩阵以及预期报酬率矩阵，这两种数据皆可以利用 proc corr 取得，接下来的各项运算，就必须依赖 SAS 所提供的矩阵运算程序，亦即 proc iml 的功能来计算，如果 R_p 为已知的数字，这样一来就可以求得在不同 R_p 下的各项资产的权重，求得权重后，投资组合的风险就会等于

$$\sigma^2 = \omega' \Sigma \omega$$

本节介绍 example_10_2.sas，该程序语法分为 2 部分，第 1 部分为整理数据到输出协方差矩阵以及预期报酬率所需要的数据，第 2 部分则利用矩阵程序撰写的宏程序来求得不同报酬率下的投资组合方差。有效边界数据整理的语句如表 10-9 所示。

表 10-9

<pre>option nonotes; libname aa 'D:\The Application of SAS in Financial Research\CH10\data'; data a; set aa.price; if y=2006 then output; if y=2005 then output; keep y m code ret; run; proc sort data=a;by code y m; run; proc means noprint data=a;</pre>
--

续表

```

var ret;by code;
output out=b(drop=_type_ _freq_) n=n;
run;
data a;
merge a b;by code;
if n=24 then output;
run;
data code;
set a;
d=1;
if code=lag(code) then delete;
keep code d;
run;
proc surveyselect data=code out=b noprint
method=srs n=20 seed=10 rep=1;
run;
data ret;
merge a b;by code;
if d=. then delete;
run;
proc sort data=ret;by y m code;
run;
proc transpose data=ret out=ret1;
var ret;
by y m;
run;
proc corr data=ret1 cov outp=b noprint;
var coll-co20;
run;
data covariance mean ;
set b;
if _type_='COV' then output covariance;
if _type_='MEAN' then output mean;
drop _type_ _name_;
run;

```

如同前一节计算多元化投资组合一样，先找出在 2005 年跟 2006 年皆存在的公司，再利用抽样程序抽出 15 家公司，在经由合并处理后，得到需要的样本公司，并且计算其平均报酬率作为预期报酬率

矩阵以及协方差。有效边界的宏语句如表 10-10 所示。

表 10-10

<pre>%macro efficient; %do i=1%to 200; proc iml; use covariance var _all_; read all into cov; use mean var _all_; read all into ret; return=&i*0.01; rp=return 1; r=(ret`) repeat(1,nrow(cov),1); w=inv(cov)*r*inv(r`*inv(cov)*r)*rp`; var=(return sqrt(w*cov*w)); varnames={ret std}; create var from var [colname=varnames]; append from var; quit; proc append data=var base=eff force; quit; %end; %mend efficient; proc datasets noprint; delete eff; quit; %efficient; options device=gif gsfname=gout xpixels=960 ypixels=720 ftext='Arial' htext=2 gunit=pct ctext=black csymbol=black; filename gout "D:\The Application of SAS in Financial Research\CH10\output\efficient.gif"; proc gplot data=eff; symbol interpol=j value=none; plot ret*std / hminor=1 vminor=1 lvref=1 cvref=black caxis=black ctext=black; run;quit;</pre>
--

程序中执行 1 到 200 共 200 次是经由试错法找出以展现有效边界图形最佳的线段，接下来逐步解析矩阵语法。将数据读入矩阵的语法如表 10-11 所示。

表 10-11

<pre>use covariance var_all_; read all into cov;</pre>
<p>use SAS table var_all_ 表示使用 SAS 文档中的所有变量，read all into matrix 表示将所有读取到的数据转成矩阵，表 10-11 所示的语法是将 covariance 这个协方差矩阵读成 cov 矩阵，mean 读成 ret 矩阵。</p>

表 10-12

<pre>return=&i*0.01; rp=return 1;</pre>
--

由于需要自行设定投资组合报酬率，但宏程序不可能执行 0.01 次，为了缩小变动单位，故将报酬率单位设为 0.01**i*，如表 10-12 所示，表示每执行一次程序是增加 0.01% 的预期投资组合报酬率，而在矩阵中，「||」指令亦为左右合并之意，若有 1 矩阵为 0.5（1 乘 1 矩阵），将之合并则为（0.5, 1）的矩阵，因此 rp 就会是（R_p, 1）。计算投资组合权重的语句如表 10-13 所示。

表 10-13

<pre>r=(ret') repeat(1,nrow(cov),1); w=inv(cov)*r*inv(r'*inv(cov)*r)*rp';</pre>
--

在表 10-13 中，nrow(cov) 为计算 cov 共有几个 row，在本例中共有 15 个，repeat（1, 15, 1）这个语句将创造出一个（1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1）的矩阵，inv 表示 inverse 之意，inv（cov）亦即为 cov⁻¹，而「'」为转置之意，经由程序的撰写，w=inv(cov)*r*inv(r'*inv(cov)*r)*rp'；便等于

$$\omega = \Sigma^{-1}(R, I)((R, I)' \Sigma^{-1}(R, I))^{-1} \begin{pmatrix} R_p \\ 1 \end{pmatrix}$$

求解投资组合的报酬率与风险的语句如表 10-14 所示。

表 10-14

<pre>var=(return sqrt(w'*cov*w)); varnames={ret std}; create var from var [colname=varnames]; append from var; quit;</pre>

算出权重之后，便计算方差，同时将方差开根号成标准偏差。Varnames={a,b} 为创造 1 个矩阵，该矩阵为（a,b），主要是为了将矩阵转存回 SAS 的 table 文档所用，create table from matrix [colname=矩阵]，由 matrix 生成一个 SAS 的 table 文档，变量名等于矩阵。将每次执行结果存储的语句如表 10-15 所示。

表 10-15

```
proc append data=var base=eff force;
quit;
```

最后将重复求得的 var 一直附加到 eff 这个 table 中，便完成这个有效边界的宏的撰写。图 10-3 为完整程序执行的结果——有效边界图。

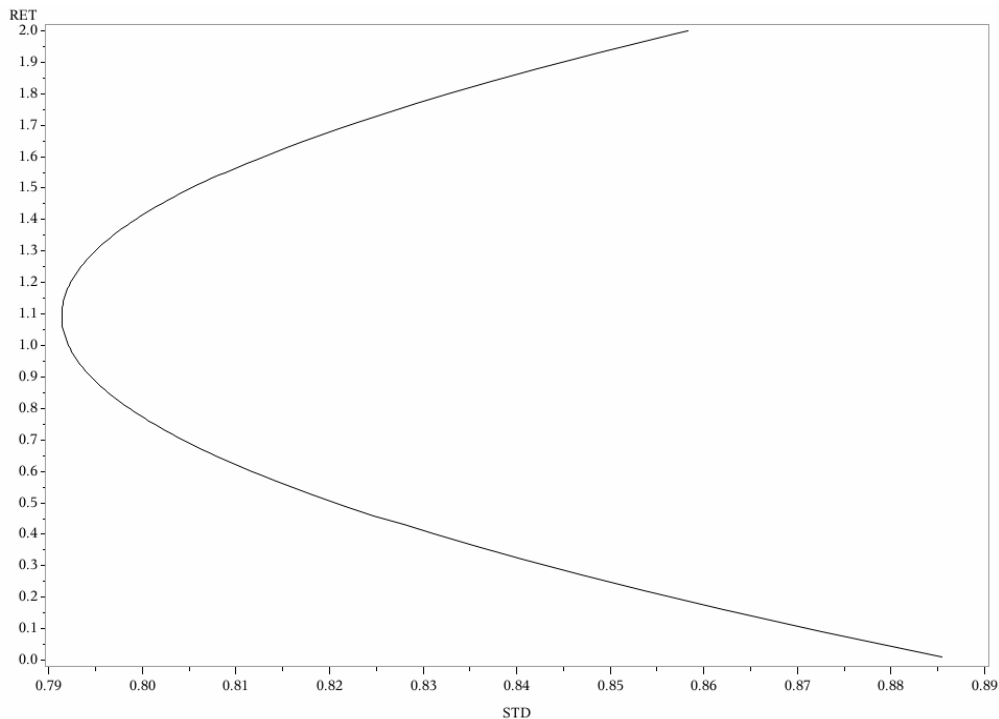


图 10-3

10.3 初探动能投资策略

在股票市场中，有许多异常现象，例如：市盈率效应、规模效应以及动能效应，从长期来看，市盈率效应指的是买进市盈率低的股票、卖出市盈率高的股票会有正报酬；而规模效应指的是买进小型公司的股票，卖出大型公司的股票在未来会有正报酬；而动能效应则只买进过去的赢家股，卖掉过去的输家股，在未来一年内会有正报酬率。

许多学术研究指出动能效应的结果在长期会有反转的现象，买进过去的赢家股，卖出过去的输家股会有负报酬率，因此衍生出两种投资策略：动能策略与反向投资策略。所谓动能策略（momentum strategy）是指买进过去的赢家股，卖出过去的输家股并且在未来会获利的一种投资方式；而反向投资

策略 (contrarian investing) 则是买进过去的输家股, 卖出过去的赢家股并且在未来会获利的一种投资方式。

DeBondt and Thaler (1985; 1987) 的研究中, 将赢家股定义为在过去表现最佳的 3 到 5 年的 50 只股票, 输家股则定义为过去表现最差的 3 到 5 年的 50 只股票, 这两位学者检验反向投资策略以验证过度反应 (overreaction) 的假设, 结果发现买进过去 5 年的输家股, 卖出过去 5 年的赢家股, 持有 3 到 5 年有显著的超额报酬。而 Jagadeesh and Titman (1993) 则发现用过去 1 年的报酬率来形成赢家与输家投资组合, 执行买进赢家卖出输家的动能投资策略会有显著的正报酬率, 值得一提的是, 和 DeBondt and Thaler 的方法不同, Jagadeesh and Titman 将股票依照过去的报酬率分成 10 组, 将报酬率最佳的股票定义为赢家, 报酬率最差的投资组合定义为输家, 在近期的研究中, 大都采用 Jagadeesh and Titman (1993) 的概念将股票分为 10 组或者其他不同的组别来进行分析, 本节以 example_10_3.sas 为例来介绍初步的动能投资策略。计算过去 J 期的复利报酬率语法如表 10-16 所示。

表 10-16

```

option nonotes nolabel;
libname aa 'D:\The Application of SAS in Financial Research\CH10\data';
data a;
set aa.price;
if mv=, then delete;
if ret=, then delete;
ret=1+0.01*ret;
month=y*100+m;
keep  code ret mv month;
run;
%macro ret(file, firm,ret,time,J,group);
proc sort data=&file;by &firm &time;
run;
data &file;
set &file;
%do i=1 %to &j;
r&i=lag&i(ret);
j&i=(geomean(of r1-r&j)-1)*100;
if &firm^=lag&i(&firm) then j&i=.;
%end;
run;
proc sort data=&file out=&file(drop=r1-r&j) ;by &time;
run;
proc rank data=&file out=&file groups=&group;

```

续表

```
var j1-j&j;  
ranks Jrank1-Jrank&j;  
by &time;  
run;  
%mend;  
%ret(a,code, ret, month, 12,10);
```

在这部分的语法中，先针对计算股票过去 1、2、3、…、12 个月的报酬率，再依照其报酬率将股票由小到大分成 10 组。取得各家公司的市值与报酬率数据如表 10-17 所示。

表 10-17

```
data ret;  
set a;  
keep month code ret mv;  
run;  
proc sort data=ret;by code month;  
run;  
data ret;  
set ret;  
mv1=mv;  
mv=lag(mv);  
if code^=lag(code) then mv=.;  
end;  
run;  
proc sort data=ret;by code descending month;  
run;  
data ret;  
set ret;  
ret1=lag(ret);  
if code^=lag(code) then ret1=.;  
run;
```

在此处制作了两个报酬率以及两个市值，其中一个是一形成投资组合以后，便立即计算报酬率，另外一个则为滞后一个月的报酬率，同时为了计算市值加权平均报酬率，我们也求得相对应的市值数据，其定义如下：

$$R_{vw,t} = \sum w_{i,t-1} r_{i,t}$$

根据定义，使用的市值为报酬率的前一期期末的市值数据，或者期初的市值数据，而这也符合实

际的状况，投资者在进行投资时，其投资权重是根据期初的数据，在使用历史数据时，会因为不小心而误用了同期的信息来计算投资组合的报酬率。在此处的整理便是根据这样的逻辑，因此对于当期的报酬率，需要抓取滞后后期的市值，而下一个月的报酬率，则使用当期的市值。抓取公司未来 60 年的月报酬率的数据如表 10-18 所示。

表 10-18

<pre>proc sort data=ret out=month(keep=month) nodupkey;by month; run; proc sort data=ret;by month; run; data month; set month; t=_n_; run; data ret; merge ret month;by month; run; data ret; set ret; do i=1 to t; if t-59<=i<=t then output; h=t-i; drop t; end; run; proc sort data=ret;by i code; run;</pre>
--

此处的数据处理是为了取得 t 月之后的 60 个月的报酬率，也就是 5 年期间的报酬率，在一般的研究或者报告中，通常会探讨 5 年的报酬率，因此要产生 5 年的数据。合并组别与报酬率的语法如表 10-19 所示。

表 10-19

<pre>data month; set month; rename t=i; run; data a; merge a month;by month;</pre>
--

续表

```
drop ret;
run;
proc sort data=a;by i code;
run;
data final;
merge a ret;by i code;
if h=, then delete;
run;
```

在此，将持有期的报酬率数据与形成期的报酬率数据分组合并在一起，这样就可以计算形成投资组合后的持有期报酬率的状况了。而在持有期报酬率方面，学术文献建议采用再平衡法，亦即如果有任意股票在持有期下市，则计算报酬率到最后一期，其余期间则以剩下的股票重新计算权重，其计算公式如下：

$$R_{(0,T)} = \prod_{t=1}^T (\sum_{i=1}^N w_{i,t-1} (1 + r_{i,t})) - 1$$

若为均等加权则 $w_{i,t-1}$ 为 $1/n$ ，根据上式定义，需要计算每一期投资组合的月报酬率，最后再计算持有投资组合的累积报酬率。求得投资组合每个月的报酬率语法如表 10-20 所示。

表 10-20

```
%macro hold(file,j,group,bit);
proc sort data=&file;by jrank&j i h;
run;
proc means noprint data=&file;
var ret l;
by jrank&j i h;
output out=ew(drop=_type_ _freq_) mean=ewr ewrl;
run;
proc means noprint data=&file;
var ret ;
weight mv;
by jrank&j i h;
output out=vw(drop=_type_ _freq_) mean=vwr ;
run;
proc means noprint data=&file;
var ret l ;
weight mv1;
by jrank&j i h;
```

续表

```
output out=vw1(drop=_type__freq_) mean=vwr1 ;
run;
proc transpose data=ew out=ewr;
var ewr;
id h;
by jrank&j i ;
run;
proc transpose data=ew out=ewr1;
var ewr1;
id h;
by jrank&j i ;
run;
proc transpose data=vw out=vwr;
var vwr;
id h;
by jrank&j i ;
run;
proc transpose data=vw1 out=vwr1;
var vwr1;
id h;
by jrank&j i ;
run;
```

首先，指定变量 file 以及 j，用来指定作为计算报酬率的文档以及过去 j 月的形成期数据，最后的 bit 则是指定进位到小数点后几位的变量，接着分别抓取每一次形成投资组合之后 60 个月的均等加权与市值加权的滞后一个月以及没有滞后一个月的报酬率。计算不同持有期的报酬率如表 10-21 所示。

表 10-21

```
data hold;
set ewr ewr1 vwr vwr1;
if jrank&j=. then delete;
j=jrank&j+1;
if nmiss(of _1-_3)=0 then k3=(geomean(of _1-_3)-1)*100;
if nmiss(of _1-_6)=0 then k6=(geomean(of _1-_6)-1)*100;
if nmiss(of _1-_9)=0 then k9=(geomean(of _1-_9)-1)*100;
if nmiss(of _1-_12)=0 then y1=(geomean(of _1-_12)-1)*100;
```

续表

```
if nmiss(of _13-_24)=0 then y2=(geomean(of _13-_24)-1)*100;
if nmiss(of _25-_36)=0 then y3=(geomean(of _25-_36)-1)*100;
if nmiss(of _37-_48)=0 then y4=(geomean(of _37-_48)-1)*100;
if nmiss(of _49-_60)=0 then y5=(geomean(of _49-_60)-1)*100;
if nmiss(of _13-_60)=0 then yh4=(geomean(of _13-_60)-1)*100;
return=_name_;
portfolio='p'|| left(j);
run;
```

在此处计算持有 3、6、9、12 个月的平均月报酬率以及持有第 2、3、4、5 年的平均月报酬率和持有第 2 年到第 5 年的平均月报酬率。现有的文献认为动能效应只存在于短期或者 1 年内，长期的报酬率则会有反转的现象，因此多以上述的持有期来探讨动能投资策略的现象。动能报酬率的检定如表 10-22 所示。

表 10-22

```
proc sort data=hold;by i return;
run;
proc transpose data=hold out=h;
var k3 k6 k9 y1-y5 yh4;
id portfolio;
by i return;
run;
data h;
set h;
mom=p&group-p1;
k=_name_;
run;
proc sort data=h;by return k;
proc means noprint data=h;
var p1-P&group mom;
by return k;
output out=b;
run;
proc transpose data=b out=b;
var p1-p10 mom;
by return k;
id _stat_;
```


续表

```

run;
data b;
set b;
t=mean/(std/(n)**0.5);
tvalue=('||left(round(t,0.01)||')');
if abs(t)>2.58 then r=round(mean, 10**-&bit)||'***';
else if 2.58>=abs(t)>1.96 then r=round(mean, 10**-&bit)||'***';
else if 1.96>=abs(t)>1.65 then r=round(mean, 10**-&bit)||'***';
else r=round(mean, 10**-&bit)||'',
run;
proc sort data=b;by return k;
run;
proc transpose data=b out=mean(drop=_name_);
var r;
by return k;
run;
proc transpose data=b out=tvalue(drop=_name_);
var tvalue;
by return k;
run;
proc transpose data=mean out=mean;
var p1-p&group mom;
by return;
id k;
run;
proc transpose data=tvalue out=tvalue;
var p1-p&group mom;
by return;
id k;
run;
data mean;
set mean;
retain t 0;
t=t+1;
type=1;
run;
data tvalue;

```

续表

```

set tvalue;
retain t 0;
t=t+1;
type=2;
run;
data j&j;
set mean tvalue;by t type;
if type=2 then _name_="";
drop t type;
run;
%mend;
%hold(final,3,10,3);
%hold(final,6,10,3);
%hold(final,9,10,3);
%hold(final,12,10,3);

```

接下来的部分为数据处理的部分，主要的目的是将数据的格式整理成学术表格报告的形式，便不多加说明，动能报酬率的检定结果如图 10-4 所示。

return	_NAME_	k3	k6	k9	y1	y2	y3
ewr	p1	0.868*	0.733**	0.736**	0.838***	1.15***	1.184*
ewr		(1.77)	(2.13)	(2.57)	(3.4)	(4.47)	(4.51)
ewr	p2	1.038**	0.918***	0.95***	1.023***	1.377***	1.258*
ewr		(2.43)	(3.07)	(3.86)	(4.75)	(5.71)	(5.12)
ewr	p3	1.316***	1.219***	1.178***	1.149***	1.298***	1.18**
ewr		(3.24)	(4.25)	(5.13)	(5.68)	(5.9)	(5.48)
ewr	p4	1.119***	1.077***	1.069***	1.108***	1.286***	1.201*
ewr		(3)	(3.96)	(4.82)	(5.7)	(6.02)	(5.46)
ewr	p5	1.25***	1.237***	1.236***	1.166***	1.134***	1.167*
ewr		(3.38)	(4.61)	(5.48)	(6.09)	(5.75)	(5.46)
ewr	p6	1.334***	1.259***	1.179***	1.102***	1.081***	1.17**
ewr		(3.64)	(4.75)	(5.5)	(5.94)	(5.45)	(5.54)
ewr	p7	1.43***	1.322***	1.283***	1.23***	1.017***	1.239*
ewr		(3.98)	(5.07)	(5.87)	(6.56)	(5.14)	(5.86)
ewr	p8	1.419***	1.319***	1.235***	1.155***	1.067***	1.106*
ewr		(3.76)	(4.91)	(5.58)	(5.98)	(5.32)	(5.37)
ewr	p9	1.338***	1.223***	1.116***	1.038***	1.07***	1.066*
ewr		(3.46)	(4.22)	(4.73)	(5.1)	(5.38)	(5.09)
ewr	p10	1.201***	1.061***	0.926***	0.849***	1.027***	1.022*
ewr		(2.79)	(3.38)	(3.67)	(3.88)	(4.87)	(4.55)
ewr	mom	0.333	0.328	0.19	0.011	-0.122	-0.162
ewr		(0.96)	(1.49)	(1.15)	(0.08)	(-0.82)	(-1.31)
ewr1	p1	0.972**	0.779**	0.791***	0.911***	1.134***	1.196*
ewr1		(1.96)	(2.25)	(2.75)	(3.74)	(4.47)	(4.55)

图 10-4

其中 mom 即是赢家减掉输家投资组合的报酬率检定，表格由上而下依序为均等加权、均等加权滞后一个月、市值加权、市值加权滞后一个月的数据，由等值加权来看，台湾地区的股市并不存在动能投资策略。

以上介绍的方法，为现有研究进行动能投资策略的方法之一，动能报酬率示意图如图 10-5 所示。

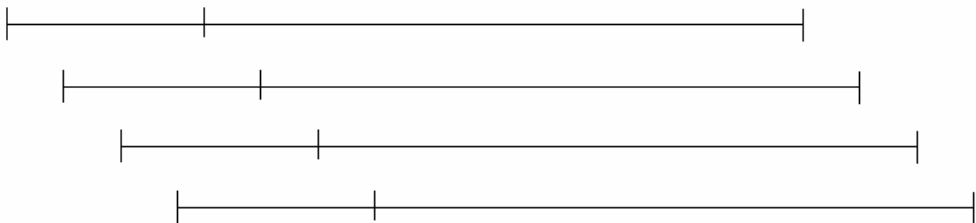


图 10-5

如图 10-5 所示，表示为每个月根据形成期的报酬率形成动能投资策略，并据此持有一段时间来检验投资策略是否能获取正报酬率，但是采用此方法，会有严重的重叠(overlapping)问题，使得投资组合报酬率会有自相关的现象，其投资组合报酬率的检定需要使用 Newey-West 调整，或者采用拔靴法(bootstrap)来进行调整。

10.4 再探动能投资策略

在前一节中，初步介绍了动能投资策略的形成方法，大多数的文献皆将其称之为 Jagadeesh and Titman(1993)提出的方法，严格来说，采用前一节的方式，只有分成 10 组这样的逻辑是符合 Jagadeesh and Titman(1993)逻辑的，这也是目前研究方面较常使用的分组方式¹，下面我们详细探讨 Jagadeesh and Titman(1993)以及 Jagadeesh and Titman(2001)的研究内容。

Jagadeesh and Titman(1993)提出了依据过去 J 个月(J months)报酬率形成投资组合，并且持有 K 个月(K months)，并将之称为 J 月/ K 月策略(J -month/ K -month strategy)，多数文献简单称其为根据过去 J 月形成投资组合并持有 K 月，在原始文献中有针对该策略做的一个详细叙述，但由于过于抽象，以至于笔者也难以理解，但在该文献的脚注 4 提到一点，原文献作者指称 12 月/12 月策略需要使用 23 个滞后期的报酬率，若依照后续文献的叙述，要使用 23 个滞后期的报酬率，应该是形成期为 23 个月，因此笔者认为有必要对此做进一步的探讨，为什么在 $J=12$ 以及 $K=12$ 的策略下，形成期的数据需要用到过去 23 个月的数据。

所幸，Jagadeesh and Titman(2001)针对 6 月/6 月策略(6 month/6 month strategy)做了一个详细的说明，在 12 月底形成的赢家投资组合是根据前 6 月到前 11 月、前 5 月到前 10 月一直到前 1 月到前 6 月的期间建构而成的，根据其说法，其投资组合的策略示意图如图 10-6 所示。

¹ 并非是一定要分成 10 组，其观念主要在于分组。

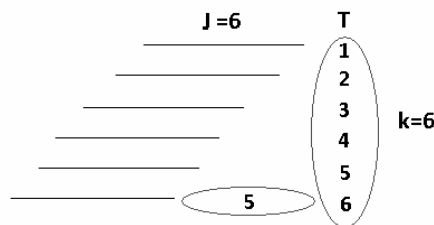


图 10-6

由图 10-6 的示意图来看，在 $J=6$ ， $K=6$ 的策略下，持有期的每个月的投资组合都有 6 个，因此在求出每一次的投资组合后，要先针对这 6 个投资组合进行均等加权平均，此时在表格计算上呈现的就是这 6 个投资组合的平均，而根据 Jagadeesh and Titman (1993; 2001) 的说法，以这种方法形成投资组合，其检定可以采用一般 t 统计量即可，接下来我们以 example_10_4.sas 为例来介绍 Jagadeesh and Titman (1993) 提出的动能投资策略检定方法，JT 动能报酬率的准备如表 10-23 所示。

表 10-23

<pre>option nonotes nolabel; libname aa 'D:\The Application of SAS in Financial Research\CH10\data'; data a; set aa.price; if mv=. then delete; if ret=. then delete; ret=1+0.01*ret; month=y*100+m; keep code ret mv month; run;</pre>

一开始，如同前一节一样准备股票报酬率的数据，包含市值以及规模这两个数据。JT 动能报酬率准备与分组语法如表 10-24 所示。

表 10-24

<pre>%macro JT(file, firm,time,J,k,group); proc sort data=&file;by &firm &time; run; data &file; set &file; %do i=1 %to &j; r&i=lag&i(ret); j&i=(geomean(of r1-r&j)-1)*100;</pre>

续表

```
if &firm^=lag&i(&firm) then j&i=.;
%end;
run;
proc sort data=&file out=&file(drop=r1-r&j) ;by &time ;
run;
proc rank data=&file out=rank(drop=j1-j&j) groups=&group;
var j&j;
ranks rank;
by &time ;
run;
```

在指定的变量中，指定了来源文档、公司码、时间码、J、K，以及要分成的组别，接着再将股票报酬率依照过去的 J 期报酬率分成数组（group）。动能报酬率分组与市值数据的整理如表 10-25 所示。

表 10-25

```
proc sort data=rank;by &firm descending &time;
data rank;
set rank;
ret=(ret-1)*100;
ret1=lag(ret);
if &firm^=lag(&firm) then ret1=.;
run;
proc sort data=rank;by &firm &time;
run;
data rank;
set rank;
mv1=mv;
rank=rank+1;
mv=lag(mv);
if &firm^=lag(&firm) then mv=.;
if rank=, then delete;
run;
```

此处先整理股票报酬率的数据，由于已经不再需要计算持有期的累积报酬率，因此先将报酬率转换成百分率的数值，接着依照上一节的方式，求得当期以及滞后一个月的报酬率和所需要的市值数据。计算每期报酬率的语法如表 10-26 所示。

表 10-26

```

proc sort data=rank ;by rank &time;
run;
proc means noprint data=rank;
var ret ret1;
by rank &time;
output out=ew(drop=_type_ _freq_) mean=ewr ewr1;
run;
proc means noprint data=rank;
var ret ;
weight mv;
by rank &time;
output out=vw(drop=_type_ _freq_) mean=vwr;
run;
proc means noprint data=rank;
var ret1 ;
weight mv1;
by rank &time;
output out=vw1(drop=_type_ _freq_) mean=vwr1;
run;
data bb;
merge ew vw vw1;by rank &time;
j=&j;
k=&k;
run;
proc sort data=rank;by &firm &time;
run;
proc datasets;
delete bbb;
run;

```

此处，先计算 $K=1$ 时的报酬率数据，接下来再依次计算其步骤所需要的数据。求得持有 K 期报酬率的语法如表 10-27 所示。

表 10-27

```

%do i=1 %to &k-1;
data b;
set rank;

```

续表

```

rank=lag&i(rank);
if code^=lag&i(code) then delete;
run;
proc sort data=b;by rank &time;
run;
proc sort data=b ;by rank &time;
run;
proc means noprint data=b;
var ret ret1;
by rank &time;
output out=ew(drop=_type__freq_) mean=ewr ewr1;
run;
proc means noprint data=b;
var ret ;
weight mv;
by rank &time;
output out=vw(drop=_type__freq_) mean=vwr;
run;
proc means noprint data=b;
var ret1 ;
weight mv1;
by rank &time;
output out=vw1(drop=_type__freq_) mean=vwr1;
run;
data bbb;
merge ew vw vw1;
j=&j;
k=&k;
run;
proc append base=bb data=bbb;
quit;
%end;

```

在此步骤，重复计算 $K=2, 3 \sim K$ 的投资组合的报酬率，以便在之后进行运算时可以使用，仔细注意一下该部分的宏语法，是要求 SAS 执行 $i=1$ 到 $K-1$ 的顺序，如果 $k=1$ ，该语法依然能够使用，看起来由 1 执行到 0 是不合理的，所以当 $K=1$ 时，SAS 将不执行该语法，而 $K=2$ 时，SAS 就执行 $k=1$ 的语法。针对 K 期报酬率进行合并的语法如表 10-28 所示。

表 10-28

```

proc sort data=bb;by j k  &time rank;
run;
proc means noprint data=bb;
var ewr ewr1 vwr vwr1;
by j k  &time rank;
output out=mean(drop=_type__freq_) mean=ew ew1 vw vw1 n=n;
run;
data mean;
set mean;
portfolio='p'|| left(rank||");
if k=n then output;
run;
proc transpose data=mean out=mean;
var ew ew1 vw vw1;
id portfolio;
by j k &time;
run;
data mean;
set mean;
mom=p&group-p1;
return=_name_;
run;
proc sort data=mean;by j k return;
run;
proc means noprint data=mean;
var p1-p&group mom;
by j  k return ;
output out=mean(drop=_type__freq_);
run;
proc transpose data=mean out=mean;
id _stat_;
by j k return;
run;
proc append base=JT data=mean;
quit;
%mend;

```

接下来将每个月得到的 K 个投资组合一起计算平均报酬率,并要求每个月的有效观测值要等于 K,

如此一来就求得了 J 月/K 月策略下下的投资组合时间序列的报酬率，接下来便针对其进行检定即可。
JT 动能宏语法的使用如表 10-29 所示。

表 10-29

```

proc datasets ;
delete jt;
quit;
%JT(a,code, month, 3,3,10);
%JT(a,code, month, 3,6,10);
%JT(a,code, month, 3,9,10);
%JT(a,code, month, 3,12,10);
%JT(a,code, month, 6,3,10);
%JT(a,code, month, 6,6,10);
%JT(a,code, month, 6,9,10);
%JT(a,code, month, 6,12,10);
%JT(a,code, month, 9,3,10);
%JT(a,code, month, 9,6,10);
%JT(a,code, month, 9,9,10);
%JT(a,code, month, 9,12,10);
%JT(a,code, month, 12,3,10);
%JT(a,code, month, 12,6,10);
%JT(a,code, month, 12,9,10);
%JT(a,code, month, 12,12,10);

```

最后计算 16 个投资组合的报酬率，并将之合并在一起，和前一节不同的是，宏完成后语法并未结束，还需要经过处理才能呈现出文献上使用的表格。JT 动能报酬率的检定如表 10-30 所示。

表 10-30

```

data b;
set jt;
t=mean/(std/(n)**0.5);
tvalue=("|left(round(t,0.01)|)");
if abs(t)>2.58 then r=round(mean, 10**-.3)||"***";
else if 2.58>=abs(t)>1.96 then r=round(mean, 10**-.3)||"**";
else if 1.96>=abs(t)>1.65 then r=round(mean, 10**-.3)||"*";
else r=round(mean, 10**-.3)||";
run;
proc sort data=b;by return j k;
run;

```

```

proc transpose data=b out=mean(drop=_name_);
var r;
by return j k;
run;
proc transpose data=b out=tvalue(drop=_name_);
var tvalue;
by return j k;
run;
proc transpose data=mean out=mean;
var p1-p&group mom;
by return j;
id k;
run;
proc transpose data=tvalue out=tvalue;
var p1-p&group mom;
by return j;
id k;
run;
data mean;
set mean;
retain t 0;
t=t+1;
type=1;
run;
data tvalue;
set tvalue;
retain t 0;
t=t+1;
type=2;
run;
data final;
set mean tvalue;by t type;
if type=2 then _name_="";
drop t type;
run;

```

JT 动能报酬率检定的结果如图 10-7 所示，使用 Jagadeesh and Titman (1993) 的 J/K 投资策略下，

台湾地区的股市不存在显著正动能报酬率,该结果与前一节类似。然而,就显著性来看,采用 Jagadeesh and Titman (1993) 的方法,其 t 统计量较前一节低, t 值确实没有膨胀的现象。

return	j	_NAME_	_3	_6	_9	_12
ew	3	p1	1.685**	1.454**	1.469**	1.411**
ew	3		(2.3)	(2.01)	(2.06)	(1.99)
ew	3	p2	1.785***	1.68**	1.613**	1.579**
ew	3		(2.69)	(2.53)	(2.46)	(2.42)
ew	3	p3	1.75***	1.749***	1.667***	1.612**
ew	3		(2.73)	(2.78)	(2.65)	(2.57)
ew	3	p4	1.756***	1.779***	1.712***	1.686***
ew	3		(2.83)	(2.86)	(2.79)	(2.74)
ew	3	p5	1.745***	1.706***	1.691***	1.719***
ew	3		(2.9)	(2.84)	(2.79)	(2.83)
ew	3	p6	1.822***	1.725***	1.721***	1.701***
ew	3		(3.15)	(2.97)	(2.93)	(2.85)
ew	3	p7	1.734***	1.596***	1.625***	1.667***
ew	3		(2.97)	(2.72)	(2.73)	(2.77)
ew	3	p8	1.639***	1.607***	1.595***	1.643***
ew	3		(2.82)	(2.75)	(2.7)	(2.74)
ew	3	p9	1.503**	1.569***	1.638***	1.709***
ew	3		(2.51)	(2.64)	(2.73)	(2.8)
ew	3	p10	1.237**	1.477**	1.614***	1.586**
ew	3		(2.04)	(2.42)	(2.59)	(2.5)
ew	3	mom	-0.449	0.023	0.145	0.175
ew	3		(-0.96)	(0.06)	(0.43)	(0.57)
ew	6	p1	1.489*	1.312*	1.211	1.328*
ew	6		(1.9)	(1.7)	(1.61)	(1.78)
ew	6	p2	1.618**	1.705**	1.658**	1.658**

图 10-7

10.5 Newey and West 的调整语法

由于动能投资组合是一时间序列的数据,在进行分析时,往往需要考虑其自相关的问题,在研究上多使用 Newey and West (1987) 提出的方法来进行方差上的调整;然而, SAS 语法中的 proc means 或者 proc univariate 中并未提供此一调整的功能,需要做一个技巧性的方法来进行调整,本节以 example_10_5.sas 为例来介绍该部分的语法。进行 AR(2) 数据的随机数据如表 10-31 所示。

表 10-31

```

option nolabel;
data a;
do t=1 to 100;
a=rannor(1);
retain y;
y=0.8*y1+0.2*y2+a;
if t=1 then do;

```

续表

```
y=a;  
y1=0;  
end;  
if t<=2 then do;  
y2=0;  
end;  
output;  
y1=y;  
y2=y1;  
end;  
run;
```

第一步，我们产生 AR (2) 的数列，该数列与前一期跟前两期都相关，由于是时间序列变量，所以要进行一些调整，使得 y1、y2 起始值由 0 开始，在 output 之后写上 y1=y，y2=y1，便指定滞后期的数值。AR (2) 数据随机数产生的结果如图 10-8 所示。

	t	a	y	y1	y2
1	1	1.8048229506	1.8048229506	0	0
2	2	-0.079915021	1.7249079297	1.8048229506	0
3	3	0.396576855	2.1214847848	1.7249079297	1.7249079297
4	4	-1.083317655	1.0381671298	2.1214847848	2.1214847848
5	5	2.2382943651	3.2764614949	1.0381671298	1.0381671298
6	6	-0.624232294	2.6522292006	3.2764614949	3.2764614949
7	7	0.5136577083	3.1658869089	2.6522292006	2.6522292006
8	8	-0.086609117	3.079277792	3.1658869089	3.1658869089
9	9	-0.594178733	2.4850990587	3.079277792	3.079277792
10	10	0.0318908181	2.5169898767	2.4850990587	2.4850990587
11	11	-0.737798572	1.7791913046	2.5169898767	2.5169898767
12	12	-0.250139175	1.5290521296	1.7791913046	1.7791913046

图 10-8

由于 proc model 并非是用来检定均值的语法，实际上是一个用来撰写特殊的实证模型的语法，而 Newey and West 的调整方法多数是用在回归模型中的，本章用了一个取巧的方法，即是利用 proc model 撰写一个回归模型¹，该模型不放入任何解释变量，因此只会估计截距项，该截距项便是被解释变量的平均值，若以 Newey and West 的方式来调整回归模型的 t 统计量，则只有在截距项的模型中，才能得到需要的结果。将数据针对时间进行排序的语法如表 10-32 所示。

表 10-32

```
proc sort data=a;by t;  
run;
```

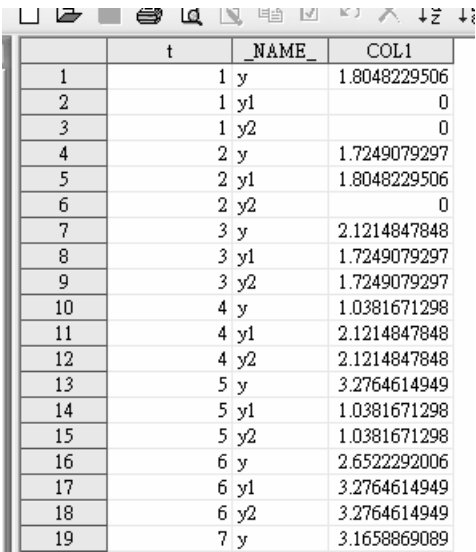
1 有关回归模型 (proc reg) 在后面的章节有更详细的介绍。

由于这是时间序列的数据，所以要先以时间顺序做排序，这一点在进行 Newey and West 方式调整回归模型时是最重要的，如果数据不按照时间顺序排序 (由大到小或由小到大皆可)，调整的方差就会有问题。将数据进行转置的语法如表 10-33 所示。

表 10-33

```
proc transpose data=a out=b;  
var y y1 y2;  
by t;  
run;
```

接下来将要检定的变量进行转置，这是因为这个程序步只能执行回归模型来得到新的统计量，而 proc model 只能跑一个回归模型，因此要将数据转换成一个数列。数据转置的结果如图 10-9 所示。



	t	_NAME_	COL1
1	1	y	1.8048229506
2	1	y1	0
3	1	y2	0
4	2	y	1.7249079297
5	2	y1	1.8048229506
6	2	y2	0
7	3	y	2.1214847848
8	3	y1	1.7249079297
9	3	y2	1.7249079297
10	4	y	1.0381671298
11	4	y1	2.1214847848
12	4	y2	2.1214847848
13	5	y	3.2764614949
14	5	y1	1.0381671298
15	5	y2	1.0381671298
16	6	y	2.6522292006
17	6	y1	3.2764614949
18	6	y2	3.2764614949
19	7	y	3.1658869089

图 10-9

以这样的数列，用户只需要针对 col1 进行分析，并且要求 SAS 依照 _name_ 变量执行程序即可，但由于要依照 _name_ 变量进行分析，需要进行排序的语法才行。依照变量时间进行排序的语法如表 10-34 所示。

表 10-34

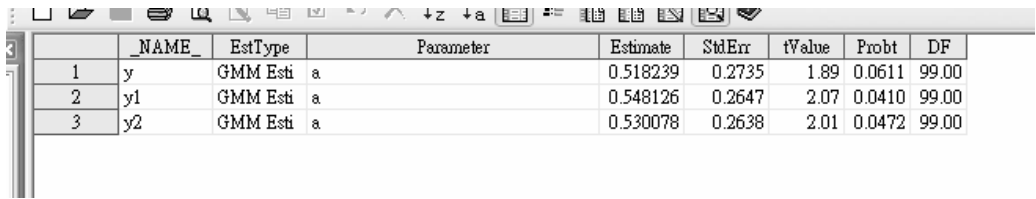
```
proc sort data=b;by _name_ t;  
run;
```

将数据整理成需要的数据格式后，就可进行分析了。Newey and West 单变量检定语法如表 10-35 所示。

表 10-35

```
%let lag=2;
PROC MODEL data=b;
PARMS a;
coll = a;
FIT coll /GMM KERNEL=(BART,%eval(&lag+1),0);
INSTRUMENTS a;
by _name_;
ods output parameterestimates=param1;
quit;
```

由于该数据是 AR（2）的数据，因此我们要调整两期，在此处使用%let lag=2 来进行，接着指定要估计的参数为 a，并且撰写 coll=a 的模型，如此估计出来的参数 a 就是我们的截距项，亦是 coll 的均值。Newey and West 检定的结果如图 10-11 所示。



	NAME	EstType	Parameter	Estimate	StdErr	tValue	Probt	DF
1	y	GMM Esti	a	0.518239	0.2735	1.89	0.0611	99.00
2	y1	GMM Esti	a	0.548126	0.2647	2.07	0.0410	99.00
3	y2	GMM Esti	a	0.530078	0.2638	2.01	0.0472	99.00

图 10-11

采用 proc means 计算相关数值的语法如表 10-36 所示。

表 10-36

```
proc means data=b mean stderr t probt;
var coll;
class _name_;
run;
```

采用 proc means 的方法一样可估计出均值、标准误、t 值以及 P 值，可以比较调整后与调整前的统计量的差异。采用一般 t 统计量的检定结果如图 10-12 所示。

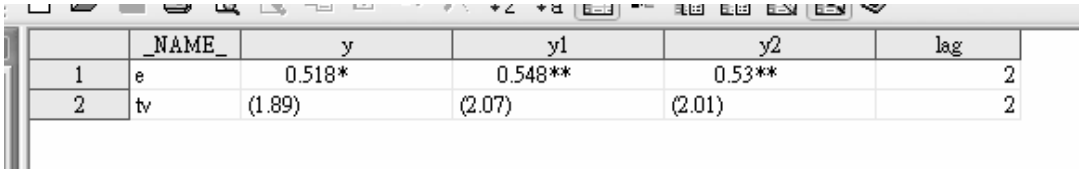
NAME	N Obs	Mean	Std Error	t Value	Pr > t
y	100	0.5182390	0.1713565	3.02	0.0032
y1	100	0.5481258	0.1677465	3.27	0.0015
y2	100	0.5300776	0.1673512	3.17	0.0020

图 10-12

结果发现未进行 Newey and West 调整的统计量较显著，接下来同样将 Newey and West 的估计结果进行表格格式化的语法，如表 10-37 所示。Newey and West 整理的结果如图 10-13 所示。

表 10-37

```
data b;
set param1;
if probt<0.01 then e=round(estimate,0.001)||'***';
else if probt<0.05 then e=round(estimate,0.001)||'**';
else if probt<0.1 then e=round(estimate,0.001)||'*';
else e=round(estimate,0.001)||'';
tv=('||left(round(tvalue,0.01)||')');
run;
proc transpose data=b out=b;
var e tv;
id _name_;
run;
data b;
set b;
lag=&lag;
run;
```



	NAME	y	y1	y2	lag
1	e	0.518*	0.548**	0.53**	2
2	tv	(1.89)	(2.07)	(2.01)	2

图 10-13

最后整理结果如图 10-13 所示，本节亦标示出调整的滞滞后期数设为几期，以方便研究者在使用后知道自己倒底做了几期的调整，接下来将该 Newey and West 的语法写成宏，如表 10-38 所示，以方便大家进行后续的分析。

表 10-38

```
%include "D:\The Application of SAS in Financial Research\CH10\program_cn\neweywest.sas";
%neweywest(a,b,y y1 y2,t,2);
%neweywest (in,out,x,time,lag);
```

在本宏中，一共设定了几个变量 in 是指来源文档，out 是指输出文档，x 是指待检定的变量，time 是指时间变量，lag 则是只要调整的期数。而时间变量并不局限于单一变量，亦可以拆分为年月日多个变量来指定，其语法如表 10-39 所示。

表 10-39

```

data a;
do t=1 to 200;
a=rannor(1);
retain y ;
y=0.8*y1+0.2*y2+a;
if t=1 then do;
y=a;
y1=0;
end;
if t<=2 then do;
y2=0;
end;
output;
y1=y;
y2=y1;
end;
run;
data a;
set a;
yy=int(t/10);
m=mod(t,10);
run;
%include "D:\The Application of SAS in Financial Research\CH10\program_cn\neweywest.sas";
%neweywest(a,b1,y y1 y2,yy m,2);

```

虽然本节并没有以投资组合的报酬率来进行检定，但也可将投资组合的报酬率利用该语法来进行调整，即可得到调整后的 t 统计量。

10.6 总结

本章介绍了投资组合股票的数目与风险以及有效边界的图形绘制，并且简单介绍了动能投资策略的程序，读者可进一步将动能投资策略的程序延伸，就可以计算规模效应、市盈率效应，甚至可以计算 Fama and French (1993) 的因子报酬率。

本章最后介绍了 Newey and West 调整的语法，该语法在财务分析上相当实用，甚至在回归语法的 Fama and MacBeth 模型中亦可使用，有兴趣的读者可以自行套用。

第 11 章

基础回归语法

在财务研究中，回归是最常使用的统计工具，使用回归方法，研究者可以得到因果关系的推论，但是更重要的是，如何经过理论的推导，选取适合的变量以及选用相对应的回归模型与回归方法。

在庞大的财务数据里，可以看到众多公司在不同时间的表现，因此需要探讨其横截面（cross-sectional）以及纵断面或者时间序列（time-series）上的回归；而如果针对某一特定产业做研究，又常会选取固定一段期间内的特定样本来检验，这时候可以采用 Panel data（面板数据）来检验固定效应（fixed effect）以及随机效应（radom effect）；另外，并非所有的研究都是属于连续变量的，在某些状况之下，研究者探讨的是二分的情形，这时候就需要纳入虚拟变量（dummy variable）来检验，甚至在探讨以虚拟变量为被解释变量时，采用 Logistic 回归模型；如果模型出现内生性问题，使得解释变量与残差项并非独立时，需要利用工具变量来进行回归，常用的为 2SLS 与 3SLS；最后再探讨两变量之间是否有领先滞后关系，或者两种投资组合之间是否有信息传递效果时，则可以采用向量自我回归模型来进行检验，在本书中，除了数据整理之外，也针对几个重要的模型进行模块化的程序撰写。

而在 SAS 之中，回归程序的指令是相当简单的，本章主要让大家了解进行回归程序所需要的数据排列方式，在以后进行数据整理时，大家便能了解未来数据整理的格式是什么。

11.1 回归语法介绍

回归分析主要是利用一组已知的变量解释或预测研究者关心的被解释变量，一般线性回归模型可以表示为下式

$$Y_i = \alpha + \beta_1 X_{1i} + \beta_2 X_{2i} + \beta_3 X_{3i} + \cdots + \beta_k X_{ki} + \epsilon_i \quad i=1,2,\cdots,N$$

其中， Y_i 是应变量、又叫做依变量或者是被解释变量 (dependent variable)， $X_{2i} \sim X_{ki}$ 为自变量、独立变量或者是解释变量 (explanatory variable or independent variable)， α 为截距项，为一个常数； $\beta_1 \sim \beta_k$ 则是估计出来的参数，为解释当 X 变动一个单位时， Y 会相对应变动几个单位的数值； ϵ 为残差，是模型无法解释的部分。

以下以 example_11_1.sas 为例来介绍基础的回归语法，如表 11-1 所示。

表 11-1

libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price';					
data a;					
set aa.price;					
run;					
proc reg data=a;					
model ret=turn;					
quit;					

在 SAS 内建的回归程序中，要进行回归语法，只要输入 proc reg 即可，同样地声明数据源就可以了，唯一不同的是，在撰写模型时，只有写 ret=turn，不同于一般线性回归模型 $ret_i = a + b * turn_i + \epsilon_i$ ，这是因为截距和残差并非是可观察的 (observable) 变量，在 SAS 中只需要输入 ret=turn，结果就等同于输入 $ret_i = a + b * turn_i + \epsilon_i$ 。回归语句 SAS 的输出结果如图 11-1 所示。

Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	1883289	1883289	6384.92	<.0001
Error	119213	35162944	294.95898		
Corrected Total	119214	37046233			
Root MSE		17.17437	R-Square	0.0508	
Dependent Mean		1.25707	Adj R-Sq	0.0508	
Coeff Var		1366.22469			
Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	1	-1.71184	0.06209	-27.57	<.0001
turn	1	0.12807	0.00160	79.91	<.0001

图 11-1

估计结果显示，截距项也就是 intercept 为-1.71184，turn 的系数值也就是 b 为 0.12807，然而模型的 R^2 只有 0.0508，表示整个模型只能解释报酬率变异的 0.0508，显示除了股票周转率之外，还有其他的解释因素，因此接下来尝试将周转率的滞后项放入模型。处理 lag 一期的周转率如表 11-2 所示。

表 11-2

```
proc sort data=a;by code y m;
run;
data b;
set a;
by code;
turn1=lag(turn);
if first.code then turn1=.;
if first.code then delete;
run;
```

首先整理样本，让数据依照公司代码、月份排序，接着另外设立一个变量为滞后期的周转率命名为 turn1，但由于数据的问题，当公司代码交替时，新公司代码的滞后期周转率数据是旧公司代码的最后一期数据，因此将之排除修正，可使用将 turn1 修正为缺失值，或者直接将数据删除。处理产生滞后一期的数据结果如图 11-2 所示。

671	2007	11	1102	-10.31	9.16	8.62
672	2007	12	1102	0.85	8.55	9.16
673	1980	2	1103	0	0.01	8.55
674	1980	3	1103	5.99	0.02	0.01
675	1980	4	1102	2.82	0.02	0.02

图 11-2

如图 11-2 所示，如果不使用 if first.code then turn1=，或者 if first.code then delete，数据就会产生错误，在进行程序分析时，务必小心。同时纳入两个回归式的语法如表 11-3 所示。

表 11-3

```
proc reg data=b;
model ret=turn;
model ret= turn1 turn;
quit;
```

由于删除了每个股票代码的第一笔数据，即使在放入了前一期周转率后，整个模型解释力的提高也未必就是由于加入滞后期周转率的原因，因此有必要再进行一次股票报酬率对股票周转率跑回归，以新的数据进行回归后的结果是 R^2 为 0.0554，较原有的数据高一点，intercept 为-1.93590，turn 为 0.12979，结果与旧数据相似。第一个回归语句的结果如图 11-3 所示。

Root MSE	16.67383	R-Square	0.0554
Dependent Mean	1.08746	Adj R-Sq	0.0554
Coeff Var	1533.28199		

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	1	-1.93590	0.06058	-31.96	<.0001
turn	1	0.12979	0.00156	83.28	<.0001

图 11-3

接下来，探讨加入滞后期的周转率会造成怎么样的结果，首先 R^2 提升到 0.1590，该结果显示新的回归模型的解释能力较佳，intercept 为-0.42270，turn1 为-0.27533，表示如果前一天或者昨天的周转率增加 1%，当期的股票报酬率就会降低-0.27533%。第二个回归语句的结果如图 11-4 所示。

Root MSE	15.73325	R-Square	0.1590
Dependent Mean	1.08746	Adj R-Sq	0.1590
Coeff Var	1446.78868		

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	1	-0.42270	0.05852	-7.22	<.0001
turn1	1	-0.27533	0.00228	-120.65	<.0001
turn	1	0.34024	0.00228	149.13	<.0001

图 11-4

事实上，股票周转率也有可能受前期报酬率的影响，投资者会关注报酬率较高的股票，因此有可能会造成交易量增加，因此先移除掉前期的股票报酬率对当期周转率的影响，亦即将周转率对滞后期的报酬率进行回归以后，取得其残差项来进行分析。取得回归结果的残差与预测值语句如表 11-4 所示。

表 11-4

```
data b;
set a;
by code;
ret1=lag(ret);
if first.code then delete;
run;
proc reg data=b noprint;
model turn= ret1;
output out=b p=pre r=res; /*取得回归模型的预测值以及残差值 分别命名为 pre res*/
quit;
```

首先同样取得报酬率的滞后期，接着进行周转率对报酬率滞后项的回归分析，由于该目标是取得

残差项的数据，所以要求 SAS 不要将回归结果输出，因此用了 noprint 这个语法。最后要求 SAS 将回归执行结果输出，可以同时取得 pre 以及 res 这两个变量。求得的残差与预测值的结果如图 11-5 所示。

	y	m	code	ret	turn	retl	pre	res
1	1980	2	1101	6.37	1.12	0.8	23.095527265	-21.97552727
2	1980	3	1101	-2.15	1.56	6.37	25.221513752	-23.66151375

图 11-5

在某些研究分析中，会采用模型的预测值作为另外一个模型的解释变量，而在另外一些研究分析中，则是采用残差值来进行的，读者可以根据自己的需要，选用适当的变量，但是在取得残差值与预测值时要注意一件事情，即由于 proc reg 可以同时进行多个模型，如果同时进行多个回归模型，但仅输入一个 output out=filename p=variablename r=variablename，则此时 SAS 仅会输出最后一个模型的预测值与残差值。如果要取得不同模型的残差值，则需要一个模型一个模型地进行回归分析，并且为不同的模型计算出来的预测值与残差值命名不同的变量名称。利用残差值进行回归分析的语法如表 11-5 所示。

表 11-5

data b;	
set b;	
resl=lag(res);	
if first.code then delete;	
run ;	
proc reg data=b;	
model ret= resl res;	
quit ;	

在取得残差值之后，可同样取得滞后项的残差值来进行回归分析，就模型而言， R^2 只有 0.1297，解释力较原来的模型降低一些，该结果并不意外，因为原有的模型中周转率包含了过去报酬率的信息，而在一些研究中，股票报酬率有自相关，亦即当期的报酬率会受过去股价报酬率的影响，当将周转率中前期报酬率的信息移除时，不可避免地会微幅下降解释能力，而结果和使用原始周转率的模型类似，周转率残差项的滞后期对当期股票报酬率有负面影响，当期的残差项则和当期报酬率之间存在正向的关系。采用残差值的回归结果如图 11-6 所示。

Root MSE	16.00235	R-Square	0.1297		
Dependent Mean	1.08707	Adj R-Sq	0.1297		
Coeff Var	1472.06635				
Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	1	1.08688	0.04655	23.35	<.0001
resl	1	-0.22904	0.00223	-102.82	<.0001
res	1	0.29482	0.00223	132.35	<.0001

图 11-6

在 SAS 的回归程序中，亦可要求进行没有截距项的分析，本质上截距项与残差项一样，都是无法被模型变量所解释的，在某些情况下，进行回归分析就必须要求进行不含截距项的回归分析，这部分将在进阶回归分析时再行探讨。采用无截距项的回归语句如表 11-6 所示。无截距项的回归语句结果如图 11-7 所示。

表 11-6

<pre>proc reg data=b; model ret= res1 res/noint; /*要求 SAS 进行没有截距项的回归模型*/ quit;</pre>					
Root MSE					
Dependent Mean					
Coeff Var					
16.03916					
1.08707					
1475.45170					
R-Square					
Adj R-Sq					
0.1292					
0.1292					
Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
res1	1	-0.22904	0.00223	-102.58	<.0001
res	1	0.29482	0.00223	132.05	<.0001

图 11-7

从模型结果可以看出，除了没有估计截距项，其余的结果与原本的估计结果是一样的，此时 SAS 只是将截距项的数值全数丢入残差项的数值中，在原有的模型所估计出来的残差项，其均值为 0，而使用无截距项的模型估计，其均值将等于旧模型的截距项，但是整体模型的解释能力与参数估计则不会有任何的变动，在 proc reg 程序中，亦可将模型依照某一变量分组进行回归分析，接下来进行分月份回归，来观察不同月份的报酬率情形。将回归系数值结果输出到 SAS 文档的语句如表 11-7 所示。

表 11-7

<pre>proc sort data=b;by m; proc reg data=b noprint outest=c; model ret= res1 res; by m; quit;</pre>					
--	--	--	--	--	--

在本例中，由于输出 1 月份到 12 月份的模型数据会产生 12 个 output 页面，为了方便起见，要求 SAS 不要将回归结果输出到 output 中，同时采用 outest 这个语法，要求将回归的系数值结果输出到 table c，回归系数值输出到 SAS 文档的程序执行结果如图 11-8 所示。

	m	_MODEL_	_TYPE_	_DEPVAR_	RMSE	Intercept	resl	res	ret
1	1	MODEL1	PARMS	ret	20.560085867	7.4026687616	-0.291779409	0.3890454093	-1
2	2	MODEL1	PARMS	ret	14.952771847	5.2684115305	-0.166400704	0.2839390703	-1
3	3	MODEL1	PARMS	ret	16.789931269	1.275877405	-0.262545428	0.2845015769	-1
4	4	MODEL1	PARMS	ret	15.590645187	0.0863418021	-0.2263524	0.2859549898	-1
5	5	MODEL1	PARMS	ret	14.04225405	-1.419686447	-0.190177021	0.2378484193	-1
6	6	MODEL1	PARMS	ret	14.34187424	-0.613779755	-0.220272835	0.2392486341	-1
7	7	MODEL1	PARMS	ret	13.988077798	-0.171919282	-0.219658021	0.3609436024	-1
8	8	MODEL1	PARMS	ret	15.130644144	-0.849768168	-0.260657984	0.2896795012	-1
9	9	MODEL1	PARMS	ret	15.60415139	-1.172384026	-0.192071194	0.2936752941	-1
10	10	MODEL1	PARMS	ret	14.106827381	-1.429517706	-0.241122147	0.2962332963	-1
11	11	MODEL1	PARMS	ret	15.679423694	1.0366331471	-0.186831068	0.2643078019	-1
12	12	MODEL1	PARMS	ret	16.793975781	3.8661748575	-0.279379239	0.3187627931	-1

图 11-8

由结果可以发现，所有的数据只显示参数估计，丝毫没有其他相关的信息，例如： R^2 或者各个参数的 t 统计量等数据，可以利用两个语法让 SAS 将这几个数据输出。要求将回归系数相关数据全部输出的语句如表 11-8 所示。

表 11-8

```
proc reg data=b noprint outest=c tableout adjrsq;
model ret=res1 res;
quit;
```

在 proc reg 的程序中，输入 tableout 以及 adjrsq 便可以将需要的数据全部存放到 table c 中，由于一次输出 12 个回归模型的数据比较繁杂，在此处仅报告一个模型的结果。完整的 SAS 回归语句的输出结果如图 11-9 所示。

	MODEL	_TYPE_	_DEPVAR_	RMSE	Intercept	resl	res	ret	_IN_
1	MODEL1	PARMS	ret	16.002354669	1.0868795149	-0.229041991	0.2948167017	-1	2
2	MODEL1	STDERR	ret	16.002354669	0.0465468023	0.0022276685	0.0022275243	.	.
3	MODEL1	T	ret	16.002354669	23.350250935	-102.8169108	132.35173605	.	.
4	MODEL1	PVALUE	ret	16.002354669	2.57043E-120	0	0	.	.
5	MODEL1	L95B	ret	16.002354669	0.9956485244	-0.233408186	0.2904507897	.	.
6	MODEL1	U95B	ret	16.002354669	1.1781105054	-0.224675796	0.2991826137	.	.

	Intercept	resl	res	ret	_IN_	_P_	_EDF_	_RSQ_	_ADJRSQ_
1	1.0868795149	-0.229041991	0.2948167017	-1	2	3	118189	0.1297006669	0.1296859397
2	0.0465468023	0.0022276685	0.0022275243
3	23.350250935	-102.8169108	132.35173605
4	2.57043E-120	0	0
5	0.9956485244	-0.233408186	0.2904507897
6	1.1781105054	-0.224675796	0.2991826137

图 11-9

到目前为止，介绍了一些回归语法的功能，接下来的部分，介绍 SAS 的参数检定以及跨模型的检定。回归语句的系数值检定语法如表 11-9 所示。

表 11-9

```
proc reg data=b;  
model ret=res1 res;  
test res1+res=1;  
test res1=1;  
test res1=res;  
test res1=-0.23;  
quit;
```

在本例子中，使用 test 来检验各个参数，需要注意的是，检定一律都只能用等式表示，此处作了四个检定，res 与 res1 的系数值加起来等于 1、res1 的系数值等于 1，res1 的系数值等于负 res，以及 res1=-0.23。回归系数值的检定结果如图 11-10 所示。

Test 1 Results for Dependent Variable ret				
Source	DF	Mean Square	F Value	Pr > F
Numerator	1	82107082	320636	<.0001
Denominator	118189	256.07535		

Test 2 Results for Dependent Variable ret				
Source	DF	Mean Square	F Value	Pr > F
Numerator	1	77947126	304391	<.0001
Denominator	118189	256.07535		

Test 3 Results for Dependent Variable ret				
Source	DF	Mean Square	F Value	Pr > F
Numerator	1	407001	1589.38	<.0001
Denominator	118189	256.07535		

Test 4 Results for Dependent Variable ret				
Source	DF	Mean Square	F Value	Pr > F
Numerator	1	47.35938	0.18	0.6672
Denominator	118189	256.07535		

图 11-10

由结果来看，前面 3 个 test 都落入拒绝区，表示前三个等式都不成立，但唯有最后一个模型显示不拒绝等式，表示 res1=-0.23 成立。

而模型间的检定则有一个重要的先决条件，等式右边的变量都要相等，即进行不同的被解释变量

与相同的解释变量的检测，接下来介绍采用数据仿真的方法来进行该部分的语法。模型间的系数检定语句如表 11-10 所示。

表 11-10

```

data a;
do i=1 to 100;
x1=rannor(1);
x2=rannor(2);
x3=rannor(2);
y1=3+2*x1+x2+3*x3+rannor(4);
y2=3+2*x1+x2+3*x3+rannor(5);
y3=2*x1+x2+3*x3+rannor(4);
output;
end;
run;
proc reg data=a;
model y1 y2 y3=x1 x2 x3;
mtest x1,x2; /*注意是逗号*/
mtest x1=2,x2=1,x3=3;
mtest y1-y2, y2 -y3, x1;
mtest y1-y2, y2 -y3, x1,x2;
mtest y1-y2;
mtest intercept;
quit;

```

首先产生 6 个变量，由于 y_1 、 y_2 、 y_3 都是进行相同的回归式，所以在模型写法上只要写上 y_1 y_2 $y_3=x_1$ x_2 x_3 ；SAS 就会自动进行 3 个回归式，而只有这样的写法，才可以进行模型检定。此处做了六个检定，第一个检定是检验三个回归模型中， $x_1=0$ ， $x_2=0$ ，在 SAS 的默认系统中，如果是检验是否等于 0，可以只打 x_1 ， x_2 ，SAS 就会认定是要检验该参数是否显著不等于 0，这部分的语法在先前介绍 test 也适用；第二个检定是检验 3 个模型的 x_1 是否等于 2、 x_2 是否等于 1、 x_3 是否等于 3；第三个检定是检验 3 个模型的 x_1 的参数是否都相同；第四个检定是检验 3 个模型中，其 x_1 与 x_2 的系数值是否相同；第五个检定则检验 y_1 与 y_2 的模型中，除了截距项，其系数值是否都一样；最后一个检定检定截距项是否相等。

在检定结果中，拒绝了第一和第二检定， x_1 与 x_2 的系数值至少有一个参数在任一个模型中不为 0，第二个检定亦是如此解释，至少有一个参数在其中的一个模型不等于所检定的等式。最后我们在检定该常数变量的系数值时，就等于检定 3 个模型的截距项是否相等，检定结果显示：3 个模型的截距项至少有 1 个和其他 3 个不相等。而进一步检定时，发现 3 个模型的截距项都不一样，事实上，模型 1 的截距项为 3.19323、模型 2 的截距项为 2.88932、模型 3 的截距项则为 -0.073。回归模型的系数检定结

果如图 11-11 所示。

Multivariate Test 1					
Multivariate Statistics and F Approximations					
	S=2	M=0	N=46		
Statistic	Value	F Value	Num DF	Den DF	Pr > F
Wilks' Lambda	0.05659635	100.37	6	188	<.0001
Pillai's Trace	0.94691122	28.47	6	190	<.0001
Hotelling-Lawley Trace	16.60700885	258.83	6	123.57	<.0001
Roy's Greatest Root	16.60327615	525.77	3	95	<.0001
NOTE: F Statistic for Roy's Greatest Root is an upper bound.					
NOTE: F Statistic for Wilks' Lambda is exact.					
Multivariate Test 2					
Multivariate Statistics and F Approximations					
	S=3	M=-0.5	N=46		
Statistic	Value	F Value	Num DF	Den DF	Pr > F
Wilks' Lambda	0.82760927	2.06	9	228.92	0.0344
Pillai's Trace	0.17837906	2.02	9	288	0.0367
Hotelling-Lawley Trace	0.20107700	2.08	9	144.56	0.0345
Roy's Greatest Root	0.15514967	4.96	3	96	0.0030
NOTE: F Statistic for Roy's Greatest Root is an upper bound.					
Multivariate Test 3					
Multivariate Statistics and Exact F Statistics					
	S=1	M=0	N=46.5		
Statistic	Value	F Value	Num DF	Den DF	Pr > F
Wilks' Lambda	0.98959617	0.50	2	95	0.6085
Pillai's Trace	0.01040383	0.50	2	95	0.6085
Hotelling-Lawley Trace	0.01051321	0.50	2	95	0.6085
Roy's Greatest Root	0.01051321	0.50	2	95	0.6085
Multivariate Test 4					
Multivariate Statistics and F Approximations					
	S=2	M=-0.5	N=46.5		
Statistic	Value	F Value	Num DF	Den DF	Pr > F
Wilks' Lambda	0.98249397	0.42	4	190	0.7933
Pillai's Trace	0.01754727	0.42	4	192	0.7906
Hotelling-Lawley Trace	0.01776365	0.42	4	112.97	0.7936
Roy's Greatest Root	0.01443164	0.69	2	96	0.5027
NOTE: F Statistic for Roy's Greatest Root is an upper bound.					
NOTE: F Statistic for Wilks' Lambda is exact.					

图 11-11

Multivariate Test 5					
Multivariate Statistics and Exact F Statistics					
S=1 M=0.5 N=47					
Statistic	Value	F Value	Num DF	Den DF	Pr > F
Wilks' Lambda	0.95316928	1.57	3	96	0.2011
Pillai's Trace	0.04683072	1.57	3	96	0.2011
Hotelling-Lawley Trace	0.04913159	1.57	3	96	0.2011
Roy's Greatest Root	0.04913159	1.57	3	96	0.2011

Multivariate Statistics and Exact F Statistics					
S=1 M=0.5 N=46					
Statistic	Value	F Value	Num DF	Den DF	Pr > F
Wilks' Lambda	0.03972257	757.47	3	94	<.0001
Pillai's Trace	0.96027743	757.47	3	94	<.0001
Hotelling-Lawley Trace	24.17460726	757.47	3	94	<.0001
Roy's Greatest Root	24.17460726	757.47	3	94	<.0001

图 11-11 (续)

回归程序步的介绍如表 11-11 所示。

表 11-11

语 法	含 义
noprint	要求不要将回归结果输出到 output
outset=tablename	将参数估计的结果输出到所命名的表格名称
tableout	详细地将参数估计结果输出到 outest 的表格中，该语法要搭配 outest 使用
adjrsq	将自由度、参数个数、R ² 以及调整后的 R ² 输出到 outest 表格中，该语法要搭配 outest 使用
model	撰写要进行的回归模型
/noint	在模型后加上 /，是指定要进行的选项语法，noint 是指不要进行截距项的估计，其他语法则可以参考 SAS 的 help
p =variablename	要求 SAS 将回归模型的预测值（残差值）输出成新的变量，在该语法前面须加上 output out=tablename
r=variablename	要求 SAS 将回归模型的残差值输出为新的变量，在该语法的前面须加上 output out=tablename
by	要求 SAS 根据 by 之后的变量分组进行回归
test	检定单一模型的参数值
mtest	检定多重模型的参数估计

11.2 格式化回归模型输出

在 SAS 中，回归模型的估计相当简便，采用 SAS 内建的程序即可估计出来，但若是估计上百个回归方程，在事后整理实证结果会产生极大的不便，因此本节使用 example_11_2.sas 来介绍回归宏语法。

首先，采用 4 因子数据来检定台湾地区股票的报酬率，以下列三式估计之。

$$R_{irf} = \alpha + \beta * R_{mrf}$$
$$R_{irf} = \alpha + \beta * R_{mrf} + s * SMB + h * HML$$
$$R_{irf} = \alpha + \beta * R_{mrf} + s * SMB + h * HML + m * MOM$$

整理因子报酬率的语法如表 11-12 所示。

表 11-12

libname aa 'D:\The Application of SAS in Financial Research\SAS data\four factor';
libname bb 'D:\The Application of SAS in Financial Research\SAS data\monthly price';
data a;
set aa.factor; run ;
data b;
set bb.price;
run ;
proc sort data=a;by y m; run ;
proc sort data=b;by y m; run ;

此处整理所需要的数据，分别将 4 因子报酬以及每月个股的股价数据读入 SAS 文档，接着按照月份排序。产生风险溢价与市场溢价的语法如表 11-13 所示。

表 11-13

data a;
merge a b;by y m;
rirf=ret-rf;
rmrf=rm-rf;
if smb=. then delete;
run ;

第二个步骤是将 4 因子数据文件以及个股报酬合并为一个文档，并且整理出回归模型所需要的个股风险溢价以及市场溢价，接下来就可以利用 SAS 进行回归模型。进行因子模型估计的语法如表 11-14 所示。

表 11-14

proc reg data=a;
model rirf=rmrf;
model rirf=rmrf smb hml;
model rirf=rmrf smb hml mom;
run ;

在 SAS 中，在写下 proc reg 之后，要再写下 data=a，这是告诉 SAS 要进行回归分析的数据取自 a 这个文档，底下则写下 3 个模型。执行 SAS 以后的 CAPM 估计结果如图 11-12 所示。

The SAS System					
2008年06月07日 星!					
The REG Procedure					
Model: MODEL1					
Dependent Variable: rirf					
Number of Observations Read			109615		
Number of Observations Used			109609		
Number of Observations with Missing Values			6		
Analysis of Variance					
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	8290657	8290657	32892.8	<.0001
Error	109607	27626543	252.05090		
Corrected Total	109608	35917200			
Root MSE		15.87611	R-Square	0.2308	
Dependent Mean		0.91636	Adj R-Sq	0.2308	
Coeff Var		1732.52517			
Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	1	0.48699	0.04801	10.14	<.0001
rmrf	1	0.94870	0.00523	181.36	<.0001

图 11-12

第一个模型 $Rirf = \alpha + \beta * Rmrf$ ，虽然在 SAS 的程序代码中，未要求 SAS 估计截距项，但是 SAS 会自行估计出来。 $Rirf = 0.487 + 0.949 * Rmrf$ 。FF 3 因子模型估计结果如图 11-13 所示。

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	1	0.07369	0.04699	1.57	0.1168
rmrf	1	0.94874	0.00513	185.07	<.0001
SMB	1	0.57580	0.00870	66.17	<.0001
HML	1	0.21807	0.00563	38.71	<.0001

图 11-13

第二个模型的估计值为 $Rirf = 0.073$ (不显著) $+ 0.949 * Rmrf + 0.576 * SMB + 0.218 * HML$ ，4 因子估计结果如图 11-14 所示。

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	1	0.09894	0.04709	2.10	0.0356
rmrf	1	0.94717	0.00513	184.66	<.0001
SMB	1	0.55609	0.00908	61.24	<.0001
HML	1	0.20945	0.00575	36.45	<.0001
MOM	1	-0.05730	0.00757	-7.57	<.0001

图 11-14

模型估计结果为 $Rirf=0.099+0.947*Rmrf+0.556*SMB+0.209*HML-0.057*MOM$ ，然而，在财务研究上由于数据特性，常常需要进行异质方差的调整，此时大多用 white (1980) 的方法进行调整。SAS 在 9.2 版本以后提供了 white 异质方差的调整，仅需要在模型之后输入 white 语法便可以输出该方差。输出 white 调整系数值的语句如表 11-15 所示。White 调整语句的结果如图 11-15 所示。

表 11-15

<pre>proc reg data=a ; model rirf=rmrf/white; model rirf=rmrf smb hml/white; model rirf=rmrf smb hml mom/white; quit;</pre>								
---Heteroscedasticity Consistent---								
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t	Standard Error	t Value	Pr > t
Intercept	1	0.09894	0.04709	2.10	0.0356	0.04567	2.17	0.0303
rmrf	1	0.94717	0.00513	184.86	<.0001	0.00638	148.42	<.0001
SMB	1	0.55609	0.00908	61.24	<.0001	0.01154	48.21	<.0001
HML	1	0.20945	0.00575	36.45	<.0001	0.00802	26.12	<.0001
MOM	1	-0.05730	0.00757	-7.57	<.0001	0.00989	-5.91	<.0001

图 11-15

由于 SAS 内建的回归输出结果仅供使用者阅读结果所用，并不能直接复制到 Word 上整理成学术文献常见的格式，故下面介绍一套宏程序，即如何将回归结果直接输出到 Excel 进行整理，学术工作者进行回归分析的目标就是要输出如下的格式，学术论文中的回归结果如图 11-16 所示。

	Regression number				
	(1)	(2)	(3)	(4)	(5)
Intercept	0.019 (.228)	0.024 (.214)	0.023 (.230)	0.010 (.549)	0.024 (.140)
Ln(sales)	0.000 (.890)	-0.001 (.839)	-0.002 (.941)	0.000 (.836)	-0.002 (.908)
ΔDiv/Price	0.154 (.699)	0.158 (.738)	0.241 (.568)	0.233 (.525)	0.238 (.387)
IA Debt/Assets	0.008 (.708)	0.008 (.711)	0.005 (.810)	0.016 (.446)	0.005 (.812)
Cash/Assets	-0.029 (.447)	-0.029 (.457)	-0.035 (.386)	-0.041 (.258)	-0.034 (.381)
Market assets/Book assets	0.004 (.479)	0.004 (.499)	0.004 (.492)	0.004 (.506)	0.004 (.468)
CEO ownership				0.055 (.134)	
CEO ownership					
Ins ownership	-0.019 (.610)	-0.006 (.860)	-0.003 (.933)		
Ins blockholdings					0.016 (.796)
Out blockholdings	0.028 (.047)			0.026 (.053)	
Corp blockholdings		0.025 (.335)			
Unaff less aff blockholdings			0.016 (.191)		0.015 (.186)
Indep board dummy	-0.019 (.029)	-0.019 (.050)	-0.020 (.028)	-0.016 (.055)	-0.020 (.024)
Poison pill dummy	-0.002 (.820)	-0.003 (.710)	-0.003 (.646)	-0.001 (.876)	-0.003 (.638)
R ²	0.076	0.063	0.067	0.083	0.067
Number of observations	173	173	173	173	173

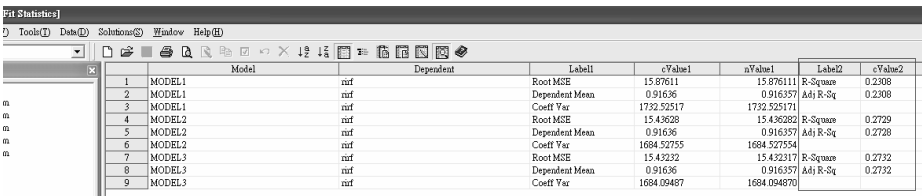
图 11-16

采用 ods 输出回归结果的语法如表 11-16 所示。

表 11-16

```
proc reg data=a ;
model rirf=rmrf smb hml mom/white ;
model rirf=rmrf /white;
model rirf=rmrf smb hml/white;
model rirf=rmrf smb hml mom/white;
ods output FitStatistics=fit;
ods output NObs=nobs;
ods output ParameterEstimates=para;
quit;
```

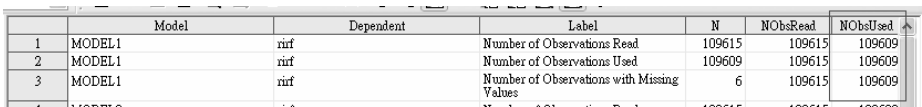
要进行回归结果的格式化程序，和相关系数一样，需要使用几个 ods 的输出表格，接下来检查这些表格的结果。fit 的表格如图 11-17 所示。



	Model	Dependent	Label	cValue1	nValue1	Label2	cValue2
1	MODEL1	rurf	Root MSE	15.87611	15.876111	R-Square	0.2308
2	MODEL1	rurf	Dependent Mean	0.91636	0.916357	Adj R-Sq	0.2308
3	MODEL1	rurf	Coeff Var	1732.52517	1732.525171		
4	MODEL2	rurf	Root MSE	15.43628	15.436282	R-Square	0.2729
5	MODEL2	rurf	Dependent Mean	0.91636	0.916357	Adj R-Sq	0.2729
6	MODEL2	rurf	Coeff Var	1684.52755	1684.527554		
7	MODEL3	rurf	Root MSE	15.43232	15.432317	R-Square	0.2732
8	MODEL3	rurf	Dependent Mean	0.91636	0.916357	Adj R-Sq	0.2732
9	MODEL3	rurf	Coeff Var	1684.09497	1684.094970		

图 11-17

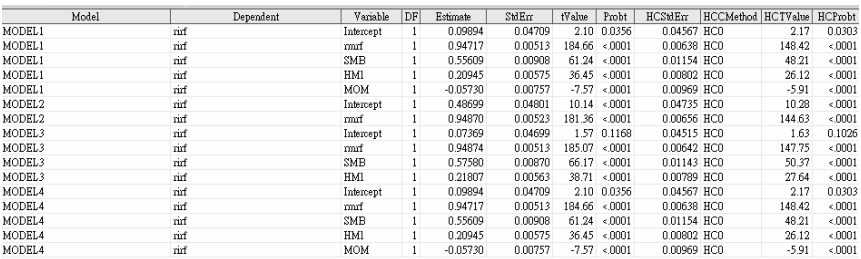
在 fit 的表格中，有学术分析需要报告的 R-Square 以及 Adj RSQ 这两个变量。Nobs 表格如图 11-18 所示。



	Model	Dependent	Label	N	NObsRead	NObsUsed
1	MODEL1	rurf	Number of Observations Read	109615	109615	109609
2	MODEL1	rurf	Number of Observations Used	109609	109615	109609
3	MODEL1	rurf	Number of Observations with Missing Values	6	109615	109609

图 11-18

在 Nnobs 中，有另外一个我们需要的变量，也就是进行回归分析所使用的观测值。接下来，我们看看 para 这个 table，如图 11-19 所示。



Model	Dependent	Variable	DF	Estimate	StdErr	tValue	Prob>	HCStdErr	HCCMethod	HCTValue	HCProb>
MODEL1	rurf	Intercept	1	0.09894	0.04709	2.10	0.0356	0.04567	HCO	2.17	0.0303
MODEL1	rurf	rmrf	1	0.94717	0.00513	184.66	<.0001	0.00638	HCO	148.42	<.0001
MODEL1	rurf	SMB	1	0.55609	0.00908	61.24	<.0001	0.01154	HCO	48.21	<.0001
MODEL1	rurf	HML	1	0.20945	0.00575	36.45	<.0001	0.00802	HCO	26.12	<.0001
MODEL1	rurf	MOM	1	-0.05730	0.00757	-7.57	<.0001	0.00969	HCO	-5.91	<.0001
MODEL2	rurf	Intercept	1	0.48699	0.04801	10.14	<.0001	0.04735	HCO	10.28	<.0001
MODEL2	rurf	rmrf	1	0.94870	0.00523	181.36	<.0001	0.00656	HCO	144.63	<.0001
MODEL3	rurf	Intercept	1	0.07369	0.04699	1.57	0.1169	0.04515	HCO	1.63	0.1026
MODEL3	rurf	rmrf	1	0.94874	0.00513	185.07	<.0001	0.00642	HCO	147.75	<.0001
MODEL3	rurf	SMB	1	0.57580	0.00870	66.17	<.0001	0.01143	HCO	50.37	<.0001
MODEL3	rurf	HML	1	0.21807	0.00563	38.71	<.0001	0.00789	HCO	27.64	<.0001
MODEL4	rurf	Intercept	1	0.09894	0.04709	2.10	0.0356	0.04567	HCO	2.17	0.0303
MODEL4	rurf	rmrf	1	0.94717	0.00513	184.66	<.0001	0.00638	HCO	148.42	<.0001
MODEL4	rurf	SMB	1	0.55609	0.00908	61.24	<.0001	0.01154	HCO	48.21	<.0001
MODEL4	rurf	HML	1	0.20945	0.00575	36.45	<.0001	0.00802	HCO	26.12	<.0001
MODEL4	rurf	MOM	1	-0.05730	0.00757	-7.57	<.0001	0.00969	HCO	-5.91	<.0001

图 11-19

Para 这个 table 有三个重要的数据，系数值、t 值以及 HCT 值（white 调整数值），在数据分析的报告表格中，通常需要报告系数值以及 t 值（HCT 值），但是这两个数据是垂直排列的，在此处却是直式水平排列的，这也是进行个性化表格程序的重要原因。接下来，开始进行个性化表格程序的编辑，以下表格型式是作者偏好输出的格式，读者若不喜欢，可以自由撰写其他的程序语法进行专属的个人表格。

在这个 OLS 宏中，我们指定了两个变量，第一个变量为 bit，表示系数值进位到小数点后几位，第二个变量则指定使用 white 调整的 t 统计量，或者是使用原始的 t 统计量。宏语法中 para 表格整理的语句如表 11-17 所示。

表 11-17

<pre>%macro OLS(bit,White); %if &white=1 %then %do; data reg; set para; if 0=<HCprobt<0.01 then sig='***'; if 0.01=<HCprobt<0.05 then sig='**'; if 0.05=<HCprobt<0.1 then sig='*'; est=round(estimate,10*(-1*&bit)) sig; tv='(' left(round(HCtvalue,0.01)) ')'; run; %end;</pre>

此处使用了 if then do; end 的宏指令用法，在这几个字前面都加上%就变成了宏语法中指定执行程序的条件语法，当我们在呼叫宏的语法中，在有关 white 的指定变量输入了 1，就会执行这一阶段的程序，其他情况则输出 ols 的结果。

首先，先针对显著星号标志进行处理，而 t 值通常只报告到小数点后 2 位，因此在程序中直接设定其为 2 位数，如果使用者希望使用 3 位数的 t 值，则可以自行修改 tv=round(HCtvalue, 0.001)，这样就可以将输出的 t 统计量改为进位到小数点后 3 位数。产生显著星号标注的语句如表 11-18 所示。

表 11-18

<pre>%else %do; data reg; set para; if 0=<probt<0.01 then sig='***'; if 0.01=<probt<0.05 then sig='**'; if 0.05=<probt<0.1 then sig='*'; est=round(estimate,10*(-1*&bit)) sig; tv='(' left(round(tvalue,0.01)) ')'; run; %end;</pre>

在这里做了另外的补充条件，即当不满足先前条件时，就执行这个部分的程序，这部分的相关 t 统计量以及 p 值是使用未调整的结果，而条件是只要不输入 white 就自动执行该语法。如果读者要按预进行的程序 1 ~ 预进行的程序 3 这三个条件来执行程序，则语法撰写如表 11-19 所示。

表 11-19

<pre>%if condition1 %then %do; 预进行的程序 1 %end; %else %if condition2 %then %do; 预进行的程序 2 %end; %else %do; 预进行的程序 3 %end;</pre>
--

接下来分别整理 R^2 以及观测值的文档，这两个文档不需要使用任何条件，只是一个直接的语法。 R^2 与观测值整理的语法如表 11-20 所示。

表 11-20

<pre>data rsq; set fit; rsq=round(nvalue2*100.02) '%'; if label2="" then delete; variable=label2; keep model variable rsq; run; data n; set nobs; variable='N'; nobs=" nobsused; if label='Number of Observations Used' then output; run;</pre>

在 R-Square 的数据中，SAS 将其区分为文本（cvalue2）以及数字（nvalue2），仅有数字数据可以计算，故使用 nvalue2 来进行 R-Square 的计算，将该数值乘以 100 在进位到小数点后两位时再和%符号合并，如果 nvalue2=0.281345 就会变为 28.13%这样的文本符号，如果读者不喜欢使用%，则可以自行做其他的变更，但切记只要改为 round(nvalue2, 0.01)||”这样的符号即可，这样才能确保数据为文本变量，在观测值的文档中，也进行相同的整理。

到目前为止，处理了每个变量的系数值、t 值、 R^2 以及观测值，请特别注意，经过整理后，这些

变量全部都变为文本变量了，最主要最后要将其放置在同一排，在 SAS 中文本变量与数字变量无法放在同一排，所以当系数值标上显著星号的那一刻起，就已经决定了所有的变量都必须转换为文本变量才行，读者在日后进行其他的语法时，也需要考虑这样的问题。模型变量的排序整理语句如表 11-21 所示。

表 11-21

<pre>data t; set reg; t=_n_; if model='MODEL 1' then output; keep variable t; proc sort data=t;by variable; run;</pre>
--

由于总共有不同模型可能会有不同的变量，这将会使得模型在排列上产生问题，因此在利用回归程序时，先让第一个模型包含之后其他模型的所有变量，接着再为这些变量进行编码，如此就可以利用编码来进行后续的工作。数据合并与输出的语句如表 11-22 所示。

表 11-22

<pre>proc sort data=t;by variable; run; proc sort data=reg;by variable; run; data reg; merge reg t;by variable; run; proc sort data=reg;by t variable; run; proc transpose data=reg out=est; var est; by t variable; id model; run; proc transpose data=reg out=tvalue; var tv; by t variable; id model; run; data OLS;</pre>

续表

```
set est tvalue;by t _name_;
if _name_='tv' then variable="";
drop t _name_;
run;
proc sort data=rsq;by variable;
run;
proc transpose data=rsq out=rsq;
var rsq;
by variable;
id model;
run;
proc transpose data=n out=n;
var nob;
by variable;
id model;
run;
data OLS;
set OLS rsq n;
%if &white=1 %then %do;
tvalue='white';
%end;
%else %do;
tvalue='OLS';
%end;
drop model1 _name_;
run;
%mend;
```

在进行完宏语法的编辑之后，就可以呼叫宏执行程序了。回归的宏语法整理如表 11-23 所示。

表 11-23

```
%ols(3,1);
proc export data=ols
outfile='D:\The Application of SAS in Financial Research\CH11\EXCELoutput\ols_result.xls'
replace;
sheet="white";
run;
%ols(3);
```

续表

```
proc export data=ols
outfile='D:\The Application of SAS in Financial Research\CH11\EXCELoutput\ols_result.xls'
replace;
sheet="ols";
run;
```

回归输出到 Excel 表格程序执行的结果如图 11-20 所示,输出到 Excel 的结果可以很轻松地整理成论文的表格¹,并且标上了系数值显著星号。

Variable	MODEL2	MODEL3	MODEL4	tvalue
Intercept	0.487*	0.074	0.099*	OLS
	(10.14)	(1.57)	(2.1)	OLS
rmrf	0.949*	0.949*	0.947*	OLS
	(181.36)	(185.07)	(184.66)	OLS
SMB		0.576*	0.556*	OLS
		(66.17)	(61.24)	OLS
HML		0.218*	0.209*	OLS
		(38.71)	(36.45)	OLS
MOM			-0.057*	OLS
			(-7.57)	OLS
Adj R-Sq	23.08%	27.28%	27.32%	OLS
R-Square	23.08%	27.28%	27.32%	OLS
N	10960	10960	10960	OLS

图 11-20

接着将数据粘贴到 Word 以后,就可以轻易快速地将回归模型结果(图 11-20)整理成如表 11-24 所示的直式 ols 整理结果,这种表格形式就比较适用于学术以及正式报告的表格。

表 11-24

	(1)	(2)	(3)
Intercept	0.487***	0.074	0.099**
	(10.14)	(1.57)	(2.1)
rmrf	0.949	0.949	0.947

¹ 基本上,对该数据进行的模型并不恰当,我们仅使用常见的回归模型来进行程序的说明。

续表

	(181.36)	(185.07)	(184.66)
SMB		0.576	0.556
		(66.17)	(61.24)
HML		0.218	0.209***
		(38.71)	(36.45)
MOM			-0.057***
			(-7.57)
R-Square	23.08%	27.28%	27.32%
AdjR-Sq	23.08%	27.28%	27.32%
N	109609	109609	109609

由于进行回归分析是经常使用的语法，而财务研究中，又常常需要进行分组回归语法，所以本书另外撰写一套分组可使用的回归三重宏语法，读者仅需要读入宏文档，就可以进行分析。回归宏语句的套装使用范例如表 11-25 所示。

表 11-25

```
/*套装化使用
关闭 SAS 都重新开启
确保 SAS 未记录 ols 宏语法
*/
%include 'D:\The Application of SAS in Financial Research\CH11\program_cn\reg.sas';
%macro model(model);
    model rirf=rmrf smb hml mom /white;
    model rirf=rmrf /white;
    model rirf=rmrf smb hml /white;
    model rirf=rmrf smb hml mom/white;
%mend;
/*
以上撰写都要跑回归的模型
*/
%reg(a,y,4);
/*
a 为数据来源文档
y 为要求 SAS 依照 y 进行回归
4 为进位到小数点后 4 位
若不分组回归 则 %a(a,4);
```

回归结果产生 reg_ols 以及 reg_white

*/

proc export data=reg_ols

outfile='D:\The Application of SAS in Financial Research\CH11\EXCELOutput\reg_result'

dbms=xlsx

replace;

sheet="ols";

run;

proc export data=reg_white

outfile='D:\The Application of SAS in Financial Research\CH11\EXCELOutput\reg_result'

dbms=xlsx

replace;

sheet="white";

run;

在套装化使用时，只需要撰写自己回归要进行的所有模型，以及指定要进行回归分析的所有变化、组别和可接受的小数点位数，就可以进行分析。

11.3 Fama-MacBeth 回归模型

在使用回归实证方法时，我们需要知道标准误差（standard errors）会导致统计推论上的问题，在先前的部分所使用的 ols 回归程序，基本上是架构在横截面回归（cross-section regression）模型中的。然而，当研究者使用横截面回归模型时，在统计推论上会忽略跨公司间的残差相关，也就是异质变异性问题；若数据是使用跨期的数据，则有可能受到数据自相关（autocorrelation）的影响（Fama and French；2002）。

在 Fama and MacBeth（1973）所开发的模型中，采用的方法是，每年分别估计出一个横截面的回归模型，接着计算出各个回归系数的平均值，并且计算出其时间序列标准误差。Fama and French（2002）指出，使用 Fama-Macbeth 回归方法所得到的系数值就像是利用面板数据（pooled time-series cross-section）回归所得到的系数值，使用 Fama-MacBeth 的方法可以捕捉到和 Panel data 估计出来的相同信息，而且获得了稳健性的标准误差（robust standard error），在进行推论上更佳。而由于 Fama-MacBeth 的方法是将回归系数看作随机变量，并检定其是否显著不为 0，在样本期间，最好超过 30 期，便能适用于 t 检定。在近期使用 Fama-MacBeth 的模型来进行的研究中，如 Hirshleifer, Hou, Teoh, and Zhang（2004）使用 462 笔的月份数据来检验投资者是否会高估在资产负债表上灌水的公司的价值，Li, Vassalou, and Xing（2006）以 1963 年第 1 季到 2000 年第 4 季共 144 笔季数据来检验产业投资成长率和股票报酬率的关系；Fama and French（2000）研究期间为 1964 年到 1996 年共 33 笔年数据来进行

公司盈余和获利性的预测模型，这些研究的数据皆符合样本期间超过 30 笔的基本假定，可以顺利地进行 Fama-MacBeth 的回归系数 t 的检定。

以下以 example_11_3.sas 为例来介绍 Fama-MacBeth 回归语法，程序中有些部分需要自行更改，其余部分则不需要更改，程序所要进行的回归模型如下式所示：

$$ret=intercept+b_1*mv+b_2*mv^2+b_3*turn+b_4*turn^2+b_5*mv_turn$$

其中 *mv* 为公司总市值，取自然对数，*mv*² 为 *mv* 的平方项，*turn* 为股票周转率，*turn*² 为 *turn* 的平方项，*mv_turn* 为 *mv* 与 *turn* 的交乘项。Fama and MacBeth 数据整理的语法如表 11-26 所示。

表 11-26

<pre>DM'LOG;CLEAR;OUT;CLEAR'; option nonotes; libname aa 'D:\The Application of SAS in Financial Research\SAS data\four factor'; libname bb 'D:\The Application of SAS in Financial Research\SAS data\monthly price'; data a; set bb.price; mv=log(mv); mv2=mv**2; turn2=turn**2; mv_turn=mv*turn; run;</pre>

首先在整理数据方面，我们重新计算 *mv*，等于取自然对数的 *mv*，在 SAS 指令中，log 为取对数的程序，其初始设定值是取自然对数，如果要以 10 为底，则应写为 log10(*mv*)，该指令就是对 *mv* 取以 10 为底的自然对数的值。*mv2=mv**2*；为计算 *mv* 的平方项，若是开根号，则指令为 *mv2=mv**(1/2)*；。接着便是进行回归。在每个时点进行横截面回归的语法如表 11-27 所示。

表 11-27

<pre>proc reg noprint adjrsq data=a outest=b ; model ret=mv turn mv2 turn2 mv_turn ; model ret=mv ; model ret=turn; model ret=mv turn; model ret=mv turn mv2; model ret=mv turn turn2; model ret=mv turn mv_turn; model ret=mv turn mv2 turn2 mv_turn ; by y m; quit; proc sort data=b;by _model_; run;</pre>

为了研究需要，分别依照月份进行了 7 个回归模型，但为了数据整理方便，此处依然增加一个含有所有变量的模型。之后再将回归结果依照模型排序，此处程序按研究所需，自行改写需要的模型，回归所要使用的数据命名为 a。程序会将回归出来的系数值输出到 b 这个文档。进行回归系数值均值的语句如表 11-28 所示。

表 11-28

```
proc means noprint data=b ;
var  intercept mv turn mv2  turn2 mv_turn _adjrsq_ ;
by _model_ ;
output out=mean(drop=_type__freq_);
run;
```

首先将估计出来的系数值（含调整后的 R^2 ）依照模型计算平均值，并输出到 mean 这个文档，此处不要求其输出特定统计量，所以 SAS 会将预设的 5 个统计量一起输出到 mean 这个文档，回归系数值的均值结果如图 11-21 所示。

MODEL	_STAT_	Intercept	mv	turn	mv2	turn2	mv_turn	_ADJRSQ_
MODEL1	N	336	336	336	336	336	336	336
MODEL1	MIN	-128.8064806	-50.84383038	-3.375027051	-3.287019156	-0.044944019	-0.533160138	-0.029159438
MODEL1	MAX	262.02894014	46.408769967	4.2866295481	2.9091121499	0.0476769863	0.4007897188	0.6676122662
MODEL1	MEAN	-0.018052692	0.0623786256	-0.165911269	0.0088560809	0.0004792687	0.0266981865	0.1340734017
MODEL1	STD	56.244514196	12.54079449	0.84316179	0.7233917698	0.0051985565	0.0973428453	0.1121477332
MODEL2	N	336	336	0	0	0	0	336
MODEL2	MIN	-87.34465194	-10.78615856	-0.012874921
MODEL2	MAX	128.94667171	12.165433442	0.4035513862
MODEL2	MEAN	-1.81909279	0.4784139692	0.0409579388
MODEL2	STD	24.189768885	2.2602974727	0.0652906252
MODEL3	N	336	0	336	0	0	0	336
MODEL3	MIN	-33.7906005	.	-0.506388441	.	.	.	-0.011861008
MODEL3	MAX	51.948881913	.	0.4251708227	.	.	.	0.5777853698
MODEL3	MEAN	0.2540335383	.	0.0655092029	.	.	.	0.062674446
MODEL3	STD	9.33617955	.	0.1347541002	.	.	.	0.0816859456

图 11-21

在 mean 的表格中，表格整理得非常清楚，同时有 _adjrsq_ 以及进行回归的月份数目，但是这些数值是以变量为横列，统计量为竖列的，所以该文档必须先进行转置，才可以做进一步的统计量的运算，回归均值结果转置宏语法的部分如表 11-29 所示。

表 11-29

```
%macro FM(bit);
proc transpose data=mean out=b;
by _model_ ;
id _stat_ ;
run;
```

此处撰写 FM (FamaMacBeth) 的宏语法，同样的，针对系数值的小数点位数作一个指定变量，接下来要求 SAS 依照模型作转置，回归均值转置的结果如图 11-22 所示。

MODEL	_NAME_	N	MIN	MAX	MEAN	STD
MODEL1	Intercept	336	-128.8064806	262.02894014	-0.018052692	56.244514196
MODEL1	mv	336	-50.84383038	46.408769967	0.0623786256	12.54079449
MODEL1	tum	336	-3.375027051	4.2866295481	-0.165911269	0.84316179
MODEL1	mv2	336	-3.287019156	2.9091121499	0.0088560809	0.7233917698
MODEL1	tum2	336	-0.044944019	0.0476769863	0.0004792687	0.0051985565
MODEL1	mv_tum	336	-0.533160138	0.4007897188	0.0266981865	0.0973428453
MODEL1	_ADJRSQ_	336	-0.029159436	0.6676122662	0.1340734017	0.1121477335
MODEL2	Intercept	336	-87.34465194	128.94667171	-1.81909279	24.189768885
MODEL2	mv	336	-10.78615856	12.165433442	0.4784139692	2.2602974727
MODEL2	tum	0
MODEL2	mv2	0
MODEL2	tum2	0
MODEL2	mv_tum	0
MODEL2	_ADJRSQ_	336	-0.012874921	0.4035513865	0.0409579389	0.0652906255
MODEL3	Intercept	336	-33.7906005	51.948881913	0.2540335383	9.33617955
MODEL3	mv	0
MODEL3	tum	336	-0.506388441	0.4251708227	0.0655092029	0.1347541002
MODEL3	mv2	0
MODEL3	tum2	0
MODEL3	mv_tum	0

图 11-22

现在表格只有参数值的均值，却没有 t 统计量，此处我们必须自行运算，要注意：t 统计量为均值除以标准误差，而标准误差可以藉由标准差除以观测值开根号求得，所以可以将数据再进行一些细节上的整理，回归系数值整理的语法如表 11-30 所示。

表 11-30

<pre>data b; set b; t=mean/(std / (n**0.5)); if t^=., then tvalue='(left(round(t,0.01)))'; if mean^=., then do; if _name_ ^='_ADJRSQ_' then do; if abs(t)>2.58 then esti=round(mean,10**-&bit) '***'; else if 2.58>=abs(t)>1.96 then esti=round(mean,10**-&bit) '***'; else if 1.96>=abs(t)>1.65 then esti=round(mean,10**-&bit) '*'; else esti=round(mean,10**-&bit) ''; end; else do; esti=round(mean*100,0.01) '%'; end; end; else do; delete; end; keep _model_ _name_ esti tvalue; run;</pre>

此处整理的重点在于使用了 if then do; else do 的语法，在交互参杂的情况下，读者可能很难操作，

要点是由最下面的 end 往上搜寻，若先对应到 do，则是一个完整的单独条件的 do end 语法，若是先遇到 end，则该语法较为复杂，多重条件语法的解释例子如表 11-31 所示。

表 11-31

if condition1 then do;(3)
if condition2 then do;(1)
运算 1
end;(1)
else do;(2)
运算 2
end;(2)
end;(3)
else do;(4)
运算 3
end;(4)

该语法是先前宏的缩小模板，分别对相呼应的 end 与 do 做了注记，end（4）对应 do（4），注记完会发现 do end（1）和（2）是 do end（3）的完整条件，该语法的解释是，在 condition 1 的条件下，如果满足 condition 2 则进行运算 1，若否则进行运算 2，最后在不满足 condition 1 的条件下，进行运算 4。因此，此处 do end（4）与 do end（3）才是完整的独立互斥条件。接下来来解读原始的语法，回归系数值的显著符号整理如表 11-32 所示。

表 11-32

if mean^=. then do;
if _name_ ^= '_ADJRSQ_' then do;(1)
if abs(t)>2.58 then esti=round(mean,10**-&bit) '***';
else if 2.58>=abs(t)>1.96 then esti=round(mean,10**-&bit) '**';
else if 1.96>=abs(t)>1.65 then esti=round(mean,10**-&bit) '*';
else esti=round(mean,10**-&bit) '';
end;(1)

在均值不为缺失值的情况下进行如下的语法，首先，如果 _name_ 不是 _adjrsq_ 就进行下面的调整，此处将系数值打上显著星号，当然若是 R² 则应该转换成百分率，因此 do end（1）将其排除掉，R² 整理的语句如表 11-33 所示。

表 11-33

else do; (2)
esti=round(mean*100,0.01) '%';
end; (2)
end; (3)

此处针对在 `_name_` 不为缺失值的情况下，且变量为 `_ADJRSQ_` 的情况下，将 R^2 换算成百分率，并且在此结束了 `do end` (2)和(3)，要注意 3 个 `do` 在此都结束了，数据删除整理的语句如表 11-34 所示。

表 11-34

<code>else do;(4)</code>
<code>delete;</code>
<code>end; (4)</code>

此处，当 `_name_` 是缺失值的状况下（即 `do (3)` 的对立条件），将数据都删掉。最后会得到如图 11-23 所示的回归均值整理结果。

<code>_MODEL_</code>	<code>_NAME_</code>	<code>tvalue</code>	<code>esti</code>
MODEL1	Intercept	(-0.01)	-0.0181
MODEL1	mv	(0.09)	0.0624
MODEL1	tum	(-3.61)	-0.1659***
MODEL1	mv2	(0.22)	0.0089
MODEL1	tum2	(1.69)	0.0005*
MODEL1	mv_tum	(5.03)	0.0267***
MODEL1	<u><code>_ADJRSQ_</code></u>	<u>(21.91)</u>	13.41%
MODEL2	Intercept	(-1.38)	-1.8191
MODEL2	mv	(3.88)	0.4784***
MODEL2	<u><code>_ADJRSQ_</code></u>	<u>(11.5)</u>	4.1%
MODEL3	Intercept	(0.5)	0.254
MODEL3	tum	(8.91)	0.0655***
MODEL3	<u><code>_ADJRSQ_</code></u>	<u>(14.06)</u>	6.27%
MODEL4	Intercept	(-0.0001)	-0.0001***

图 11-23

结果基本上跟上一节 `ols` 的做法一样，但是在这边仍然需要进行一些调整，因为研究者并不需要 R^2 的 `t` 值，因此需要对它进行调整，对系数与 `t` 统计量再整理的语句如表 11-35 所示。

表 11-35

<code>data t;</code>
<code>set b;</code>
<code>t=_n_;</code>
<code>if _model_='MODEL1' then output;</code>
<code>keep _name_ t;</code>
<code>run;</code>
<code>proc sort data=t;by _name_;</code>
<code>run;</code>
<code>proc sort data=b;by _name_;</code>
<code>run;</code>
<code>data b;</code>
<code>merge b t;by _name_;</code>

续表

```
run;

proc sort data=b;by t _name_;

run;

proc transpose data=b out=esti;

var esti;

id _model_;

by t _name_;

run;

proc transpose data=b out=tvalue;

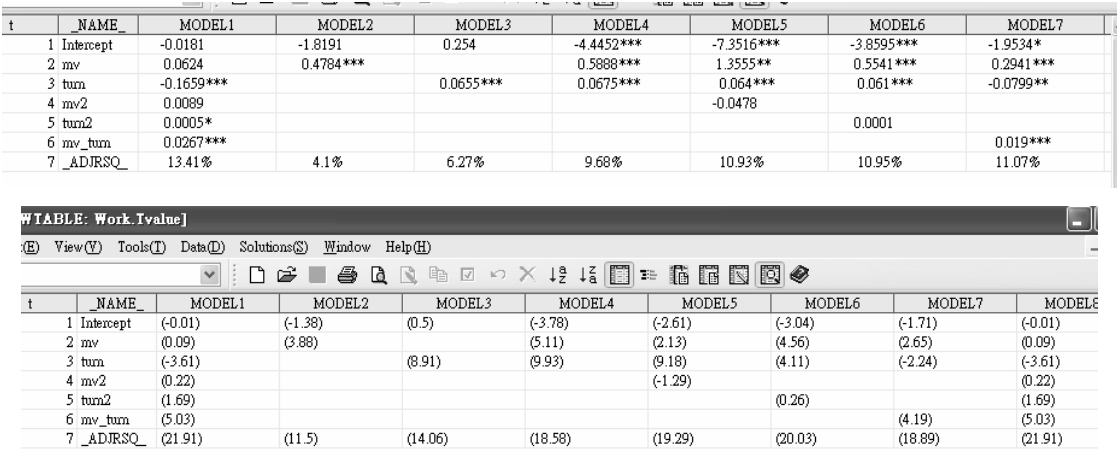
var tvalue;

id _model_;

by t _name_;

run;
```

同前一节 ols 的处理一样，可以得到如图 11-24 所示的回归系数与 t 统计量输出结果的两个文档。



t	_NAME_	MODEL1	MODEL2	MODEL3	MODEL4	MODEL5	MODEL6	MODEL7
1	Intercept	-0.0181	-1.8191	0.254	-4.4452***	-7.3516***	-3.8595***	-1.9534*
2	mv	0.0624	0.4784***		0.5888***	1.3555**	0.5541***	0.2941***
3	turn	-0.1659***		0.0655***	0.0675***	0.064***	0.061***	-0.0799**
4	mv2	0.0089				-0.0478		
5	turn2	0.0005*					0.0001	
6	mv_turn	0.0267***						0.019***
7	_ADJRSQ_	13.41%	4.1%	6.27%	9.68%	10.93%	10.95%	11.07%

t	_NAME_	MODEL1	MODEL2	MODEL3	MODEL4	MODEL5	MODEL6	MODEL7	MODEL8
1	Intercept	(-0.01)	(-1.38)	(0.5)	(-3.78)	(-2.61)	(-3.04)	(-1.71)	(-0.01)
2	mv	(0.09)	(3.88)		(5.11)	(2.13)	(4.56)	(2.65)	(0.09)
3	turn	(-3.61)		(8.91)	(9.93)	(9.18)	(4.11)	(-2.24)	(-3.61)
4	mv2	(0.22)				(-1.29)			(0.22)
5	turn2	(1.69)					(0.26)		(1.69)
6	mv_turn	(5.03)						(4.19)	(5.03)
7	_ADJRSQ_	(21.91)	(11.5)	(14.06)	(18.58)	(19.29)	(20.03)	(18.89)	(21.91)

图 11-24

在 tvalue 这个文档中，有 7 个观测值，而在 t=7 时是不需要 R^2 的，因此可以利用 SAS 内建的观测值总数的语法将其删除，系数值与 t 统计量整理的语法如表 11-36 所示。

表 11-36

```
data esti;

set esti;

type=1;
```

续表

```
run;  
data tvalue;  
set tvalue nobs=x;  
if t=x then delete;  
type=2;  
_name_="";  
run;
```

将不需要的数据删除，并且设立一个变量 type，使系数值与 t 值呈现上下的顺序，系数值与 t 统计量合并输出 Excel 的语法如表 11-37 所示。Fama and MacBeth 回归结果如图 11-25 所示。

表 11-37

```
data FM;  
set esti tvalue;by t type;  
drop t type model1;  
run;  
%mend;  
%fm(4);  
proc export data=FM  
outfile='D:\The Application of SAS in Financial Research\CH11\EXCELoutput\FM'  
dbms=xlsx  
replace;  
sheet='FM1';  
run;
```

NAME	MODEL2	MODEL3	MODEL4	MODEL5	MODEL6	MODEL7	MODEL8
Intercept	-1.8191 (-1.38)	0.254 (0.5)	-4.4452*** (-3.78)	-7.3516*** (-2.61)	-3.8595*** (-3.04)	-1.9534* (-1.71)	-0.0181 (-0.01)
mv	0.4784*** (3.88)		0.5888*** (5.11)	1.3555** (2.13)	0.5541*** (4.56)	0.2941*** (2.65)	0.0624 (0.09)
tum		0.0655*** (8.91)	0.0675*** (9.93)	0.064*** (9.18)	0.061*** (4.11)	-0.0799** (-2.24)	-0.1659*** (-3.61)
mv2				-0.0478 (-1.29)			0.0089 (0.22)
tum2					0.0001 (0.26)		0.0005* (1.69)
mv_tum						0.019*** (4.19)	0.0267*** (5.03)
ADJRSQ	4.1 %	6.27 %	9.68 %	10.93 %	10.95 %	11.07 %	13.41 %

图 11-25

如此，得到所需要的回归结果了。和前一节一样，整理完数据之后，就可以快速地粘贴到如表 11-38 所示的直式 Fama-MacBeth 整理结果表格了。

表 11-38

variable	(1)	(2)	(3)	(4)	(5)	(6)	(7)
intercept	-1.8191	0.254	-4.4452***	-7.3516***	-3.8595***	-1.9534*	-0.0181
	(-1.38)	(0.5)	(-3.78)	(-2.61)	(-3.04)	(-1.71)	(-0.01)
mv	0.4784***		0.5888***	1.3555**	0.5541***	0.2941***	0.0624
	(3.88)		(5.11)	(2.13)	(4.56)	(2.65)	(0.09)
turn		0.0655***	0.0675***	0.064***	0.061***	-0.0799**	-0.1659***
		(8.91)	(9.93)	(9.18)	(4.11)	(-2.24)	(-3.61)
mv ²				-0.0478			0.0089
				(-1.29)			(0.22)
turn ²					0.0001		0.0005*
					(0.26)		(1.69)
mv_turn						0.019***	0.0267***
						(4.19)	(5.03)
adjrsq_	4.1%	6.27%	9.68%	10.93%	10.95%	11.07%	13.41%

有关表格个人化整理的程序，笔者亦将其整理存储成另外的 FM 程序文件，读者可自行加以使用，Fama and MacBeth 宏语法使用范例如表 11-39 所示。

表 11-39

```

proc reg noprint adjrsq data=a outest=b ;
  model ret=mv turn mv2 turn2 mv_turn ;
  model ret=mv ;
  model ret=turn;
  model ret=mv turn;
  model ret=mv turn mv2;
  model ret=mv turn turn2;
  model ret=mv turn mv_turn;
  model ret=mv turn mv2 turn2 mv_turn ;
  by y m;
quit;
proc sort data=b;by _model_;
run;
proc means noprint data=b ;
  var intercept mv turn mv2 turn2 mv_turn _adjrsq_;
  by _model_;
  output out=mean(drop=_type_ _freq_);

```

续表

```
run;
%include 'D:\The Application of SAS in Financial Research\CH11\program_cn\FM.sas';
%fm(4);
proc export data=FM
outfile='D:\The Application of SAS in Financial Research\CH11\EXCELOutput\FM2'
dbms=xlsw
replace;
sheet='FM';
run;
```

读者在撰写程序时，要自行将需要的模型一一撰写，并且将数据的均值输出到 mean，相关变量的计算请依照所使用的变量估计。

在前面的章节中，已经介绍了采用 Newey and West 的方式来调整投资组合报酬率的方差，在 Fama and MacBeth 的回归方程检定可以使用相同的方式加以调整，本书亦撰写了该语法的宏，example_11_4.sas 的内容如下。

(1) Fama and MacBeth 回归的语句如表 11-40 所示。

表 11-40

```
DM'LOG;CLEAR;OUT;CLEAR';
option nonotes nlabel;
libname aa 'D:\The Application of SAS in Financial Research\SAS data\four factor';
libname bb 'D:\The Application of SAS in Financial Research\SAS data\monthly price';
data a;
set bb.price;
mv=log(mv);
mv2=mv**2;
turn2=turn**2;
mv_turn=mv*turn;
run;
proc sort data=a;by y m;
run;
proc reg noprint adjrsq data=a outest=b ;
model ret=mv turn mv2 turn2 mv_turn ;
model ret=mv ;
model ret=turn;
model ret=mv turn;
model ret=mv turn mv2;
```

续表

```
model ret=mv turn turn2;
model ret=mv turn mv_turn;
model ret=mv turn mv2 turn2 mv_turn;
by y m;
quit;
```

一开始依照相同的方式撰写每个时期进行一个回归，接下来直接使用已经写好的调整语法。
(2) 使用 Newey and West 检定 Fama and MacBeth 系数的范例如表 11-41 所示。

表 11-41

```
%include "D:\The Application of SAS in Financial Research\CH11\program_cn\FMneweywest.sas";
%fmneweywest(b,a, intercept mv turn mv2 turn2 mv_turn, y m,4,2);
%fmneweywest(in,out, x, time,bit,lag);
/*
in 是指要进行 neweywest 的文档
out 是指要输出的文档名称
x 是指所有的变量名称，包含截距项
time 是指时间变量
bit 是指四舍五入到小数点的后几位
lag 是指要考虑多少及它的滞后项，若为 0，就是一般的 white 调整
*/
```

使用该宏须指定几个变量，in 是指来源文档，out 是指输出文档，x 是针对要检定的回归变量，time 是时间变量，bit 是指系数值要四舍五入到小数点后的位数，lag 则是要检定的期数，经过简单的处理之后，就可以得到 Newey and West 调整后的 Fama and MacBeth 回归系数的检定了。

11.4 总结

本章介绍了回归程序语法，并且简单地介绍了如何将 SAS 的回归结果个人化输出到 Excel 文档中，还介绍了 Fama MacBeth 的做法，也可将进行个人化表格的程序另外存储成 reg 以及 FM，有需要的读者可以自行取用。

第 12 章

回归语法的应用

第 11 章简单介绍了回归的基础语法，并且提供了两种格式化表格的输出语法，虽然介绍了一些较复杂的方法来做格式化表格，但是这些回归方法本质上都只是采用横截面分析方法，本章则着重有关时间序列上的应用，借由财务研究上常用的 CAPM、3 因子以及 4 因子模型的估计，介绍移动窗口（moving window）、滚动数据法（rolling），并进而介绍结构性改变（structural change）以及分段回归模型（piecewise regression）。

严格来说，CAPM、3 因子以及 4 因子模型并不难估计，但是许多研究生仍对这部分程序语法感觉很难理解，主要的原因在于虽然单一时点，估计特定股票的 CAPM、3 因子以及 4 因子模型并不难估计，但是如果采用 Fama and French（1992）的做法，在每年 6 月底时，使用每家上市公司过去 60 个月并要求至少存在 24 个月估计所得的数据来作为各家公司的 α 、 β 以及其他共同因子（common factor）的因素负荷量（factor loading）就有些困难了。

要采用 Fama and French（1992）的方法来估计因子模型的话，由于这部分的数据牵涉到不同公司、不同时间，对于初学者而言，乍看起来似乎有些困难，但是如果巧妙地运用 proc sort 以及 by 这两个程序及其语法，整个程序语法的估计只在于如何在每年 6 月底抓取过去 60 个月的数据进行所有公司的回归，本章首先介绍移动窗口的方法，并借助该方法教授大家如何解决这个棘手的问题。

12.1 移动窗口 (moving window)

本节主要介绍财务研究经常使用的窗口，事实上所谓窗口，可以认为是研究测试期间，所谓移动窗口，指的是每次研究的期间长度不变，就跟窗户一样，可以将窗户开合移动，虽然窗户的位置改变了，但是窗户的宽度不变。以财务数据为例，假设研究者开一窗口为 60 个月，总样本期间为 1981 年 1 月到 1986 年 12 月，一开始窗口期间为 1981 年 1 月到 1985 年 12 月，共 60 个月，接下来移动 1 个月，窗口期间为 1981 年 2 月到 1986 年 1 月，共 60 个月，……最后一个窗口为 1982 年 1 月到 1986 年 12 月，虽然研究期间为 1981 年 1 月到 1986 年 12 月共 72 个月，但是藉由移动窗口的做法，我们产生了 13 个 60 个月的样本。如果总样本其间变长，研究者观察到的不同位置的窗口也就越多，移动窗口本质上就是增加样本数的方法。撰写移动窗口的程序有两种写法，一种写法是采用宏语法，另一种写法则是采用 data set 的方法。宏语法会重复读取原始数据，并一期一期地估计，比较耗时，但是在程序撰写上，可以节省硬盘空间；data set 的方法，如果硬盘容量太小则不建议使用，以下使用 example_12_1.sas 来介绍该语法。股价数据与大盘报酬率数据整理的语句如表 12-1 所示。

表 12-1

```
DM'LOG;CLEAR;OUT;CLEAR';
option nonotes nolabel;
libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price';
libname bb 'D:\The Application of SAS in Financial Research\SAS data\four factor';
data a;
set aa.price;
keep code y m ret ;
run;
data rm;
set bb.factor;
t=_n_;
run;
proc sort data=a;by y m;
run;
```

首先准备股票报酬率数据以及因子数据，因子数据包含大盘报酬率（rm）、无风险利率（rf）以及 SMB、HML 和 MOM，由于因子数据是每月一笔数据，因此可以很轻易地对每个月做编码的动作，事实上当研究者需要对股票交易日作编码的动作时，可以直接抓取大盘报酬率的数据，因为只要大盘没有事务数据，个股也一定不会有数据，而个股则有可能临时停止交易，因此在因子数据中我们加入 t=_n_ 的语法，由于因子数据是根据份额由小到大排序的，所有其观测值顺序便可作为每月的编码，最小为 1，最大为 306。因子报酬率整理的语句如表 12-2 所示。

表 12-2

```

data a;
merge a rm;by y m;
rmrf=rm-rf;
rirf=ret-rf;
keep code y m t rirf rmrf smb hml mom;
run;

```

接着将股票报酬率数据和因子数据合并，并计算 rirf 及 rmrf，最后再保留 CAPM、3 因子以及 4 因子模型所需要的数据，到此为止，数据的准备工作就算完成了。

移动窗口回归的宏语法如表 12-3 所示。

表 12-3

```

%macro moving_window;
proc datasets;
delete final;
run;
%do i=60 %to 306;
data b;
set a;
if &i-59<=t<=&i then output;
run;
proc sort data=b;by code;
run;
proc reg data=b outest=b adjrsq noprint;
model rirf=rmrf;
model rirf=rmrf smb hml;
model rirf=rmrf smb hml mom;
by code ;
quit;
data b;
set b;
t=&i;
if _p_+_edf_>=24 then output;/*要求至少存在 24 个月*/
keep code t _model_ intercept rmrf smb hml mom;
run;
proc append base=final data=b;
quit;
%end;

```

```

data t;
set rm;
keep y m t;
run;
data final;
merge final t;by t;
if t<60 then delete;
run;
%mend moving_window;
%moving_window;

```

在宏语法中，先删除掉 final 的 table，事实上该语法是最后才写进宏程序的，该语法的写入是为了避免重复执行程序时，在宏最后的语法中的附加程序（proc append）会将旧的数据跟新数据合并，当读者阅读到这方面的语法时，要先了解该指令是最后下的，并且之后会使用到 proc append。

接着，要求 SAS 由 60 重复执行语法到 306，要求 SAS 将本期之前（含本期）60 个月的月数据存入 table b 中，然后将数据依照股票代码排序，就可以进行三个模型的回归估计，而有趣的是 _p+_edf_ 这个语法，这两个变量必须在回归程序中，下半部分的 adjrsq 这个语法，其中 _P_ 是回归模型的参数个数，_edf_ 则为自由度，两者相加便是回归中的有效样本数目，根据 Fama and French（1992）的说法，有效样本数目至少要 24 个月，因此要求 SAS 将有效样本数大于 24 的模型结果输出，接着将该回归结果全部合并并在 final 这个 table 中，而在回归结果中，仅能保留 t 这个数据，却已经遗失掉月份的信息，因此重新设一个 table t，读取 rm 这个 table，在保留月份以及 t 这些变量后和 final 合并。请读者注意：此处必须要将因子数据删除，因为在回归结果中，系数值名称和原始因子数据一样，合并之后会影响原有系数的结果，而在最后合并之后删除掉前 60 个月的数据，就取得了样本期间内第 60 个月到第 306 个月的移动窗口的回归结果。

然而，在程序撰写过程中，仅仅是要求 SAS 将过去 60 期的数据都输出，但是如果有一只股票在 24 个月前就终止上市了，在估计数据时，就该舍去这部分的数据，因此还需要考虑每家公司在样本期间内最后的时点，否则会产生一些多余而不可靠的数据。取得每家公司在样本期间最后月份的语法如表 12-4 所示。

表 12-4

```

proc sort data=a out=code (keep=code t);by code descending t;
run;
data code;
set code; by code;
endt=t;
if first.code then output;
drop t;
run;

```

利用有完整数据的 a 文档，要求 SAS 将其依照 code，以及由大到小排列 t，这样一来，每只股票最后的 t 便是在每只股票的第一个观测值，重新设立一新变量 endt，再要求 SAS 将 code 的每一笔数据输出，便拥有每只股票在样本期间的最后一个月的信息了，用同样的方法，也可以取得最早出现的第一笔观测值。合并整理出需要的系数值的语法如表 12-5 所示。

表 12-5

<pre>proc sort data=final;by code; run; data final; merge final code;by code; if t=, then delete; if t<=endt then output; drop t endt; run; data final66; set final; if m=6 then output; run;</pre>
--

接下来要求 SAS 仅输出 $t \leq \text{endt}$ 的数据，如此便可以保证每只股票的回归模型只进行到其样本期间的最后一个月。

介绍完宏语法的移动窗口之后，接下来介绍 data set 的方法，该方法在整理数据部分与宏程序相同，唯一有所改变的是在宏程序这部分的语法，采用 data set 进行移动窗口的语法如表 12-6 所示。

表 12-6

<pre>data b; set a; do i=1 to t; if t-59<=i<=t then output; end; run; proc sort data=b;by code i; run; proc reg data=b outest=b adjrsq noprint; model rir=rmr; model rir=rmr smb hml; model rir=rmr smb hml mom; by code i; quit;</pre>

```

data b;
set b;
t=i+59;
if _p_+_edf_>=24 then output;
keep code t _model_ intercept rmrf smb hml mom;
run;
proc sort data=b;by t;
run;
data t;
set rm;
keep y m t;
run;
data final;
merge b t;by t;
if t<60 then delete;
run;
proc sort data=a out=code (keep=code t);by code descending t;
run;
data code;
set code; by code;
endt=t;
if first.code then output;
drop t;
run;
proc sort data=final;by code;
run;
data final;
merge final code;by code;
if t=. then delete;
if t<=endt then output;
drop t endt;
run;

```

首先，取出文档 b，要求读取 a，并且 do i=1 to t，SAS 会重复产生 t 的最后一个数据，在本例中为 306 个，并且有一个新的变量 i。

这种移动窗口方式的数据形式如图 12-1 所示，当 t 等于 1 时，会产生 i=1 的一笔数据，当 t=2 时，会产生 i=1、2 的相同的该笔数据，如果我们要进行三期的移动窗口语法，则当 i 等于 1 时，输出 t=1

到 3 的数据, $i=2$ 时, 输出 $t=2$ 到 4 的数据, 亦即如果要进行 n 期的移动窗口, 我们要找出的是 $t-n+1 \leq i \leq t$ 的数据, 因此要输出 60 期的数据, 就要下一个指令, 要求当 i 介于 $t-59$ 与 t 之间便会输出, 这部分的语法可以和前面的宏语法相呼应, 但完全是相似的做法。

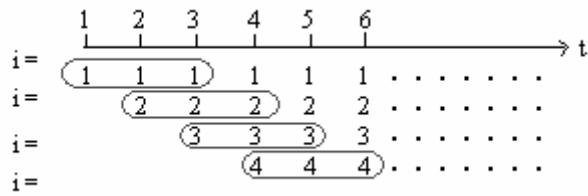


图 12-1

如此将所有的数据都读取完毕以后, 就可以进行三种模型的因子回归, 接下来的做法便与宏程序语法相似, 执行完程序语法之后, 就可以求得所有月份的移动窗口的因子模型系数值。

行文至此, 读者可能有所怀疑, 一开始的目的是输出每年 6 月的回归系数值, 现在则有每个月根据过去 5 年至少存在 2 年的回归系数, 用户该如何处理呢? 取得特定月份数据的语法如表 12-7 所示。

表 12-7

<code>data final6;</code>
<code>set final;</code>
<code>if m=6 then output;</code>
<code>run;</code>

只要利用前面章节所介绍的观测值输出法, 输出每年月份等于 6 的数据, 就完成了财务研究上经常使用的计算 CAPM、3 因子模型与 4 因子模型的数据。此处并非是直接针对每年 6 月去抓取要进行回归的数据, 反之, 本书推荐先计算所有月份的回归结果, 最后再输出 6 月份的回归结果, 这种做法虽然比较繁杂, 但对于研究者而言, 藉由这种方法可以建立起其回归数据库, 而不用每次在进行研究时, 都要重新计算一次了。

12.2 共同基金绩效评估：移动窗口的应用

本节藉由衡量共同基金的绩效来介绍移动窗口程序的特殊应用, 衡量共同基金绩效的方法有以下几种:

第一种为 Treynor (1965) 所提出的 Treynor ratio, 其认为共同基金绩效衡量为风险溢价除以系统风险, 其定义如下:

$$Treynor\ ratio_i = (E(R_i) - R_f) / \beta_i$$

第二种为 Sharpe (1966) 提出的 Sharpe ratio, 其认为绩效衡量的方式应为共同基金的风险溢价除

以其所承担的总风险，其定义如下：

$$Sharpe\ ratio_i = (E(R_i) - R_f) / \sigma_i$$

第三种为 Jensen (1968) 提出的 Jensen's alpha，亦即 CAPM 模型中的截距项，其定义如下：

$$r_{irf_i} = \alpha_i + \beta_i \times rmrf$$

以上三种皆为常见的基金绩效衡量方式，其着重的一点是共同基金的择股能力，而有后续学者提出共同基金经理人除了择股能力之外，还有择时能力，学术研究中最常用来衡量共同基金经理人的择时能力共有两种。

第一种为 Treynor and Mazuy (1966) 所提出的，其认为共同基金经理人能够预测市场趋势，在市场绩效较佳时，会增加投资组合中的股票，其系统风险 β 会较高；当期预测市场绩效较差时，会降低投资组合中的股票，其系统风险 β 会较低。因此提出了以下的式子来衡量基金经理人的择时能力：

$$r_{irf_i} = \alpha_i + \beta_{1i} \times rmrf + \beta_{2i} \times rmrf^2 + \varepsilon_i$$

其中， β_{1i} 为传统 CAPM 的系统风险的因素负荷量， β_{2i} 则是指基金经理人的择时能力。

第二种为 Henriksson and Merton (1981) 提出的，如果基金经理人具有预测市场的能力，其会跟着市场状况而调整其系统风险的大小，其回归模型如下：

$$r_{irf_i} = \alpha_i + \beta_i \times rmrf + \gamma_i \times rmrf \times D_i + \varepsilon_i$$

上式中， D_i 为一虚拟变量，当 $rmrf > 0$ 时为 1，其他为 0，而 γ 就是基金经理人的择时能力。

在介绍完以上五种基金绩效衡量方法之后，接下来以 example_12_2.sas 为例将这五种绩效一起估计出来。

共同基金数据整理的语法如表 12-8 所示。

表 12-8

<pre>libname aa 'I:\The Application of SAS in Financial Research\SAS data\fund data'; libname bb 'I:\The Application of SAS in Financial Research\SAS data\four factor'; data rm; set bb.factor; if y<1987 then delete; drop smb hml mom mturn; run;</pre>

由表 12-8 可知，首先第一步取得样本所需要的数据，由于共同基金数据最早始自 1987 年，因此我们在抓取因子报酬率数据时，将 1987 年以前的数据全部删除。

共同基金大盘报酬率整理的语句如表 12-9 所示。

表 12-9

<pre>data rm; set rm; t=_n_;</pre>

续表

```
run;
proc sort data=rm;by y m;
run;
data a;
merge rm aa.fundret;by y m;
rirf=ret-rf;
rmrf=rm-rf;
TM=rmrf*2;
if rmrf>=0 then HM=rmrf; else HM=0;
if rirf=., then delete;
run;
proc sort data=a;by fund;
run;
```

同样的步骤，针对所有的月份做编码，接着再将因子数据与共同基金数据进行水平合并¹，由于择时能力有两种模型，此处分别将变量命名为 TM 与 HM，以检验在样本期间中基金是否具有择时能力。共同基金绩效的宏语句如表 12-10 所示。

表 12-10

```
%let year=2;
%macro a;
proc datasets;
delete      fund_performance;
quit;
%do i=&year*12 %to 252;
data b;
set a;
if      &i-(&year)*12+1<=t<=&i then output;
run;
proc sort data=b;by name;
run;
proc means noprint data=b;
var rirf;
by name;
output out=c(drop=_type_ _freq_) mean=Erirf std=std n=n;
/*计算预期的风险溢价以及标准偏差*/
```

1 关于共同基金数据已经事先处理过，共有 5 只基金，并且依照月份做好排序。

```

run;

proc reg noprint data=b outest=d;
model rirf=rurf;
by name;
quit;

data fund;
merge c d;by name;
Trenor=Erirf/rurf;
Sharpe=Erirf/std;
Jensen=intercept;
t=&i;
if n=&year*12 then output;
keep name trenor sharpe jensen t;
run;

proc append base=fund_performance data=fund;
quit;

proc sort;by t;
%end;

data fund_performance;
merge fund_performance rm ;by t;
if m=12 then output; /*若不要只是年底的数据则可以删除之*/
run;

proc sort ;by name;
proc means mean t;
var trenor sharpe jensen;
class name;
run;
%mend a;
%a;

```

在宏程序前声明了%let year=2; 这个语法, 是因为本宏程序是以年度为主的, 范例中已由&year*12 将其转换为月数据, 如果有非年度数据, 例如 11 个月或 13 个月来进行程序运行的话, 则可自行将 &year*12 加以修改。此外, 在程序中加入了 if m=12 then output; 这个语法, 表示在进行分析时, 仅计算年底的数据, 如果想要有更多的样本数据, 则可以将其删除。年底时期共同基金的绩效结果如图 12-2 所示。

name	Obs	Variable	Mean	t Value
汇丰台湾	17	Treynor	0.2462306	0.41
		Sharpe	0.0144945	0.40
		Jensen	-0.5063706	-1.83
汇丰安富	16	Treynor	0.3534519	0.81
		Sharpe	0.0623285	1.48
		Jensen	0.2602711	1.79
汇丰基金	18	Treynor	0.0248448	0.05
		Sharpe	0.0079931	0.17
		Jensen	-0.2414003	-0.90
汇丰万邦	17	Treynor	4.7512502	0.31
		Sharpe	-0.1095474	-1.89
		Jensen	-0.6619604	-1.81
汇丰龙凤	12	Treynor	0.6769649	2.01
		Sharpe	0.1003195	1.98
		Jensen	0.6881941	3.17

图 12-2

由图 12-2 所示的结果发现，除了汇丰龙凤基金之外，其他 4 只基金的绩效皆不佳，如果想要申购基金的话，汇丰龙凤基金似乎是较佳的选择，除了仅输出单月份的数据外，也针对每个月的数据进行分析，全时期共同基金绩效的结果如图 12-3 所示。

name	Obs	Variable	Mean	t Value
汇丰台湾	193	Treynor	-0.3378798	-1.61
		Sharpe	-0.0101521	-0.80
		Jensen	-0.5371135	-7.98
汇丰安富	181	Treynor	0.2678349	1.88
		Sharpe	0.0560368	3.81
		Jensen	0.2268424	4.48
汇丰基金	208	Treynor	-0.2536319	-2.07
		Sharpe	-0.0096529	-0.86
		Jensen	-0.2595068	-3.58
汇丰万邦	193	Treynor	-21.1609761	-3.93
		Sharpe	-0.1112821	-7.15
		Jensen	-0.6793147	-6.60
汇丰龙凤	193	Treynor	0.7229297	6.94
		Sharpe	0.1042542	7.68
		Jensen	0.6996690	11.83

图 12-3

由图 12-3 可见，所示的结果不变，汇丰龙凤基金的绩效依然是 5 只基金中绩效表现最佳的基金，而汇丰安富基金也具有不错的绩效，其余 3 只共同基金的绩效甚至比不上无风险利率，在探讨完这三种绩效之后，我们进一步观察基金的择时能力。

共同基金择时绩效的回归语句如表 12-11 所示。

表 12-11

```
proc reg data=a;
model rirf=rmrf TM;
model rirf=rmrf HM;
by name;
quit;
```

简单的执行回归之后，即可取得 5 只基金的择时能力，在此不报告该部分的 SAS 执行结果，有兴趣的读者可以自行执行程序检验该结果，本例子的结果显示共同基金不存在择时能力，因为所有的择时回归系数皆不显著。

12.3 滚动法（Rolling）

本节主要介绍财务研究上的滚动法，事实上所谓滚动法，可以看作是移动窗口的延伸，所谓移动窗口，指的是每次研究的期间长度不变，而滚动法则是每次样本都增加一期，就跟滚雪球一样，当提供雪球足够的能量之后，雪球就会自动地从山上滚动下去，并且会越滚越大，以财务数据加以说明，假设研究至少要有 60 个月的样本，总样本期间为 1981 年 1 月到 1986 年 12 月，一开始期间为 1981 年 1 月到 1985 年 12 月，共 60 个月，接下来我们滚动 1 个月，期间为 1981 年 1 月到 1986 年 1 月，共 61 个月，……，最后一个窗口为 1980 年 1 月到 1986 年 12 个月，总共 72 个月，我们产生了 13 个样本。这些样本会越来越大，和移动窗口的固定观察月数不一样。撰写滚动法的程序和移动窗口一样，有两种写法，一种是采用宏语法，另一种则是采用 data set 的方法。因此，本质上滚动式语法与移动窗口的程序是大同小异的，接下来以 example_12_3.sas 为例来介绍滚动式语法。

滚动回归宏语句如表 12-12 所示。

表 12-12

```
DM'LOG;CLEAR;OUT;CLEAR';
option nonotes nlabel;
libname aa 'I:\The Application of SAS in Financial Research\SAS data\monthly price';
libname bb 'I:\The Application of SAS in Financial Research\SAS data\four factor';
data a;
set aa.price;
keep code y m ret ;
run;
data rm;
set bb.factor;
t=_n_;
```

续表

```

run;
proc sort data=a;by y m;
run;
data a;
merge a rm;by y m;
rmrf=rm-rf;
rirf=ret-rf;
keep code y m t rirf rmrf smb hml mom;
run;
%macro rolling;
proc datasets;
delete final;
run;
%do i=60 %to 306;
data b;
set a;
if t<=&i then output; /*此处与移动窗口语法不同*/
run;
proc sort data=b;by code;
run;
proc reg data=b outest=b adjrsq noprint;
model rirf=rmrf;
model rirf=rmrf smb hml;
model rirf=rmrf smb hml mom;
by code ;
quit;
data b;
set b;
t=&i;
if _p_+_edf_>=24 then output; /*为了和移动式窗口比较，设为 24，可改为 60*/
keep code t _model_ intercept rmrf smb hml mom;
run;
proc append base=final data=b;
quit;
%end;
data t;
set rm;

```

续表

```
keep y m t;
run;
data final;
merge final t;by t;
if t<60 then delete;
run;
%mend rolling;
%rolling;
```

基本上，宏程序仅仅修改了一小部分，在移动窗口中为 if &i-59<=t<=&i then output，而在此处则删除了&i-59 的部分。

在使用 data set 的方法之前，先思考一个问题，在移动窗口的部分，使用 do i=1 to t；这个语法可以顺利地将移动窗口的语法撰写出来，但是在滚动式语法中，这个方法是否就没有办法进行了呢？我们使用滚动回归时期示意的图 12-4 作为图例介绍。

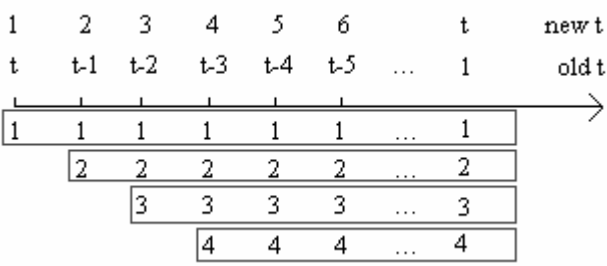


图 12-4

如果将数据反序列排序，就可以顺利地利用 do i=1 to t 的方式，进行滚动的是回归语法，在移动窗口中，其开始的起始点与结束的终止点不同，但是可以藉由条件句调整不同的终止点，在滚动式语法中，其起始点都是一样的，因此反过来针对终止点来撰写程序，在撰写程序的时候，可以试着在纸上画出数据的形式，利用图像来帮助自己思考程序的撰写。

回归数据处理的语句如表 12-13 所示。

表 12-13

```
DM'LOG;CLEAR;OUT;CLEAR';
option nonotes nlabel;
libname aa 'I:\The Application of SAS in Financial Research\SAS data\monthly price';
libname bb 'I:\The Application of SAS in Financial Research\SAS data\four factor';
data a;
set aa.price;
```

续表

```
keep code y m ret ;  
run;  
data rm;  
set bb.factor;  
tt=_n_;  
run;  
proc sort data=rm;by decending tt;  
data rm;  
set rm;  
t=_n_;  
run;  
proc sort data=rm;by y m;  
run;
```

此处的语法相当简便，针对时间的变量做了两次排序，最主要是因为滚动式语法是从最后一期往回推的。而在之后的语法中，不会针对样本的期间再加以修正，读者可以思考其中的原因。

数据步滚动式回归整理的语法如表 12-14 所示。

表 12-14

```
proc sort data=rm;by y m;  
run;  
proc sort data=a;by y m;  
run;  
data a;  
merge a rm;by y m;  
rmrf=rm-rf;  
rirf=ret-rf;  
if t=, then delete;  
keep code y m t rirf rmrf smb hml mom;  
run;  
data b;  
set a;  
do i=1 to t;  
output;  
end;  
run;  
proc sort data=b;by code i;
```

续表

```
run;
proc reg data=b outest=b adjrsq noprint;
model rirf=rmrf;
model rirf=rmrf smb hml;
model rirf=rmrf smb hml mom;
by code i;
quit;
```

该部分的语法几乎与移动窗口相同，只是使用的是所有的数据，其原因如前面的图所示，就是需要所有的数据才行，因此在此处不需要调整数据。

滚动式回归语法如表 12-15 所示。

表 12-15

```
proc reg data=b outest=b adjrsq noprint;
model rirf=rmrf;
model rirf=rmrf smb hml;
model rirf=rmrf smb hml mom;
by code i;
quit;
data b;
set b;
t=i;
if _p_+_edf_>=24 then output;
keep code t _model_ intercept rmrf smb hml mom;
run;
proc sort data=b;by t;
run;
data t;
set rm;
keep y m t;
run;
data final;
merge b t;by t;
if t<60 then delete;
run;
data final66;
set final;
if m=6 then output;
run;
```


至此，将数据完全地计算出来，读者可以自行比较两种语法的程序结果是否相同，当然，移动与滚动式回归语法有时会要求计算下一期的残差值，藉以计算模型的预测能力，其做法是将当期的回归系数值与下一期的数据合并：

$$\varepsilon_{t+1} = Y_{t+1} - \hat{\alpha}_t - \hat{\beta}_t X_{t+1}$$

利用该方法，可以将每一期的残差值都计算完成，接着根据研究目的，可以做进一步的检验，而事实上财务研究中的事件研究法，亦是采用相似的概念来计算异常报酬率的。

12.4 Where 语法有妙招

本节新增介绍一个 SAS 中实用的语法，可以大大地节省移动平均以及滚动平均法处理数据上的难处，在前面的介绍中，采取先将数据读取的方法，然后再进行 proc means 以及 proc reg 的处理，实际上 SAS 提供了一个有用的语法，可以让用户在读取数据集时，先行筛选需要的数据，并且可节省 SAS 运行时读取数据的时间，以下以 example_12_4.sas 为例来介绍。

采用条件式进行移动窗口回归的语句，如表 12-16 所示。

表 12-16

<pre>proc means noprint data=a(where=(&i-(&year)*12+1<=t<=&i)); var rirf; by name; output out=c(drop=_type_ _freq_) mean=Erirf std=std n=n; /*计算预期的风险溢价以及标准偏差*/ run; proc reg noprint data=a(where=(&i-(&year)*12+1<=t<=&i)) outest=d; model rirf=rirf; by name; quit;</pre>
--

不管是在 proc means 或者 proc reg 中，在 data=a 之后下了一个指令(where= (&i-(&year)*12+1<=t<=&i))，这是声明在进行描述统计或者跑回归时，采用的数据要满足这个条件，t 要介于 &i-(&year)*12+1 以及 &i 之间，亦即不需要让 SAS 先读取新的文档%do i=&year*12 %to 252；若进行的数据量很大时，采用这个步骤可更有效率地执行移动窗口的结果。移动窗口与采用 where 语法后的回归语句的比较如表 12-17 所示。

表 12-17

<pre>data b; set a; if &i-(&year)*12+1<=t<=&i then output; run; proc sort data=b;by name; run; proc means noprint data=b; var rirf; by name; output out=c(drop=_type__freq_) mean=Erirf std=std n=n; /*计算预期的风险溢价以及标准偏差*/ run; proc reg noprint data=b outest=d; model rirf=rmrf; by name; quit;</pre>	<pre>proc means noprint data=a(where=(&i-(&year)*12+1<=t<=&i)); var rirf; by name; output out=c(drop=_type__freq_) mean=Erirf std=std n=n; /*计算预期的风险溢价以及标准偏差*/ run; proc reg noprint data=a(where=(&i-(&year)*12+1<=t<=&i)) outest=d; model rirf=rmrf; by name; quit;</pre>
--	---

接下来说明如何利用 where 来修改滚动法的回归语法，滚动窗口与采用 where 语法后的回归比较如表 12-18 所示。

表 12-18

<pre>data b; set a; if t<=&i then output; /*此处与移动窗口语法不同*/ run; proc sort data=b;by code; run; proc reg data=b outest=b adjrsq noprint; model rirf=rmrf; model rirf=rmrf smb hml; model rirf=rmrf smb hml mom; by code ; quit;</pre>	<pre>proc reg data=a(where=(t<=&i)) outest=b adjrsq noprint; model rirf=rmrf; model rirf=rmrf smb hml; model rirf=rmrf smb hml mom; by code ; quit;</pre>
---	--

利用这个方法，不需要让 SAS 先行读取数据，而是每一次回归时，都直接要求 SAS 读取我们需要的数值，虽然在采用月的数据时，where 的功能无法一下子显现出来，但如果是使用日的数据，其功能则能很快显现。我们在进行财务实证时，往往会遇到如下的情况。

“本研究在 t 年 6 月底时采用 t-1 年 7 月份到 t 年 6 月份所有的日的数据来求得其日报酬率均值、标准差以及求得 CAPM 模型下的 β 值”，该说明亦是一种移动窗口的做法，虽然其移动的年份都是 1 年，月份都是 12 个月，但是每年的交易日数据都是不一样的，而且采用的是跨越两个年度的数据，但若使用 where 语法再加上第 2 章介绍的时间函数，就可以处理了。采用条件读取文档的语句如表 12-19 所示。

表 12-19

<code>proc means data=a(where=(mdy(7,1,&i-1)<=t<=mdy(6,30,&i)));</code>
<code>proc means data=a(where=(mdy(7,1,&i-1)<=mdy(m,d,y)<=mdy(6,30,&i)));</code>

利用 where 加上时间函数，用户可以很方便地写出论文要求的语法，因为不管每一年交易日有几天，这所有的交易日的时间格式都是介于(mdy(7, 1, &i-1)以及 mdy(6, 30,&i)之间的，所以读者只要利用每一天的年 (y)、月 (m)、日 (d)，就可以计算出每一年需要的结果。

12.5 结构性改变

在财务研究中，由于常会收集到跨期或者不同公司类型的数据，将使得回归模型不能够很好地适配，例如，台湾地区证券集中市场为使市场交易与揭示制度更公平、更有效率、信息更透明，自 2002 年 7 月 1 日起实施新制度，实施项目分为四项：（1）盘中搓合取消两档限制、（2）盘中瞬间价格稳定措施、（3）收盘改采用 5 分钟集合竞价、（4）增加揭露未成交之买卖委托价量信息。这样一来，台湾地区股票市场的报酬率与交易量结构或许会因此改变，因为如果将 2002 年 7 月以前的数据与 2002 年 7 月以后的数据视为相同结构，以报酬率对周转率进行回归模型，或许会有些许不妥；学术文献 Morck, Shleifer and Vishny（1998）提出了当管理者的持股与公司价值有结构上改变的问题，即当管理者持股低于 5%时，持股比率与公司价值呈现正相关，介于 5%到 25%时，管理者持股比率与公司价值呈现负相关，高于 25%时，管理者持股比率与公司价值又呈现正相关。

上述两个例子分别指出，如果对公司数据适配相同的回归模型，会使得回归模型以及回归结果不稳定，因此需要将数据区隔开来并且适配不同的模型。一般而言，检验数据是否有结构性改变的方法如下，将数据依照产生结构性改变的点分成两个次样本，分别进行回归模型，利用其残差平方和与总样本的残差平方和进行 F 检定，也就是常见的 Chow test。

有关 Chow test 的检定步骤如下。

第一步，针对所有完整样本估计一条回归式，取得其残差平方和。

$$y_f = \sum_{i=1}^k \beta_i x_i + \varepsilon_f$$
$$ESS_f = \sum_{j=1}^N \varepsilon_{fj}^2$$

第二步，将样本依照所探讨的结构性改变的点分为两段，重新估计相同的模型，分别取得其残差平方和：

$$y_j = \sum_{i=1}^k \beta_i x_i + \varepsilon_j \quad j=1,2$$

$$ESS_1 = \sum_{j=1}^{N_1} \varepsilon_{1j}^2 \quad ESS_2 = \sum_{j=N_1+1}^{N_2} \varepsilon_{2j}^2 \quad N = N_1 + N_2$$

第三步：计算 Chow test 的 F 分数

$$F_{(k, N-2k)} = ((ESS_f - ESS_1 - ESS_2) / k) / (ESS_1 + ESS_2) / (N - 2k)$$

其中， k 为模型估计的参数个数， N 为样本的总观测值。

以下使用台湾地区股票市场的报酬率与周转率来检定 2002 年 7 月以后是否产生结构性改变，该语法如 example_12_5.sas 所示。

设计回归断点的语法如表 12-20 所示。

表 12-20

<pre>libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price'; data a; set aa.price; month=y*100+m; if month>=200207 then change=1;else change=0; keep ret turn change; run;</pre>
--

首先将数据进行整理，先设定虚拟变量 Change，当月份大于 2002 年 7 月时为 1，其他为 0。分段进行回归的语句如表 12-21 所示。

表 12-21

<pre>proc sort data=a;by change; run; proc reg data=a noprint outest=k(keep=_p_) adjrsq; model ret=turn; output out=a r=r1; run; proc reg data=a noprint ; model ret=turn; by change; output out=a r=r2; run;</pre>

在该部分中，将数据依照 change 排序，接着进行全部数据的回归，将该部分回归模型的残差输出变量命名为 r1，并同时要求 SAS 将参数估计的结果输出到 k 文档，并下了 adjrsq 的指令，实际上该部分是要取得参数个数变量 p_，所以我们只保留 p_ 变量以作为之后合并所用。

接着再进行同样的回归模型，但此时要求 SAS 依照 change 变量分别进行回归，要求 SAS 将该部分残差输出变量命名为 r2。

求得回归断点所需残差值平方和的语句如表 12-22 所示。

表 12-22

<pre>data a; set a; ESS1=r1**2; ESS2=r2**2; run; proc means noprint; var ESS1 ESS2; output out=b(drop=_type_ _freq_) sum=ESS1 ESS2 n=N; run;</pre>
--

在表 12-22 这部分程序中，先将之前回归所得的残差值求出其平方值，接着利用 proc means 这个程序，分别计算其残差值的平方和，并且将有效观测值输出，如此一来，得到了所有计算 Chow test 的数据，ESS1 为 ESS_r ，而 ESS2 则为 ESS_1+ESS_2 ，_p_的数据为 k，而 N 的数值即为 N。计算 Chow test 的语句如表 12-23 所示。

表 12-23

<pre>data b; merge b k; Chow=((ESS1-ESS2)/_p_)/(ESS2/(N-2*_p_)); ProbF=1-probf(Chow,_p_,N-2*_p_); keep Chow ProbF; run; proc print data=b; quit;</pre>
--

在将数据计算出 Chow test 所需的 F 分数外，我们同时求出其 p-value。

先前的部分，是在已知结构点可能发生的时点，直接针对该时点设定虚拟变量分段进行回归检定即可，但是若只是知道数据有结构性改变，却不知道发生的时点时，则需要采用另外的方法进行。产生结构改变的数据语句如表 12-24 所示。

表 12-24

<pre>data a; do t=1 to 100; x=rannor(1); y=2+3*x+rannor(2); output;</pre>

续表

```
end;
run;
data a;
set a;
do i=2 to 99;
if i<=t then change=1;else change=0;
output;
end;
run;
proc sort data=a;by i;
run;
```

首先产生一笔随机数据，假设其为时间序列的数据，并且重制 98 次。并要求设定虚拟变量，i 小于 t 时设虚拟变量为 1，其他为 0。之后依照 i 排序，可以得到 i=2 时，t=1 为 0，其他为 1；i=3 时，t=1，2 为 0，其他为 1。

	t	x	y	i	change
1	1	1.804822951	7.334553831	2	0
2	2	0.396576855	2.10641291	2	1
3	3	2.238294365	8.090650801	2	1
4	4	0.513657708	3.454364008	2	1
...

	t	x	y	i	change
100	100	1.754990224	8.943564546	2	1
101	1	1.804822951	7.334553831	3	0
102	2	0.396576855	2.10641291	3	0
103	3	2.238294365	8.090650801	3	1
...

图 12-5

检验某个时间点产生结构性改变语句的语法如表 12-25 所示。

表 12-25

```
proc reg data=a noprint adjrsq outest=k(keep=i _p_);
model y=x;
by i;
output out=a r=r1;
quit;
proc reg data=a noprint adjrsq;
model y=x;
```

续表

```
by i change;
output out=a r=r2;
quit;
data a;
set a;
ESS1=r1**2;
ESS2=r2**2;
run;
proc means noprint;
var ESS1 ESS2;
by i;
output out=b(drop=_type_ _freq_) sum=ESS1 ESS2 n=N;
run;
data b;
merge b k;by i;
Chow=((ESS1-ESS2)/_p_)/(ESS2/(N-2*_p_));
ProbF=1-probf(Chow,_p_,N-2*_p_);
if probf<0.1 then output;
keep i Chow ProbF;
run;
proc print data=b;
quit;
```

接下来的语法仅仅是修正先前的部分，由程序依照每个 i 算出其是否有结构性改变，为简便起见，在最后的部份只要求输出 P-value 达 10%显著水平的数据。Chow test 检验结果如图 12-6 所示。

The SAS System				
Obs	i	Chow	ProbF	
1	92	2.73695	0.069825	
2	96	2.44859	0.091797	
3	97	2.77288	0.067492	
4	98	2.42259	0.094096	

图 12-6

结果发现在 i=92 开始有结构性改变，但 P-value 为 0.06，且将样本分为 91 笔与 9 笔，意义也不大，如果采用更严谨的做法，则显著水平可采用 P-value 为 0.05，因此本数据可看成结构没有改变。

12.6 分段回归 (piecewise regression)

虽然在前一节中，发现数据并没有发生结构性改变，但该结果是由于随机数生成的，因此没有结构性改变的结果并不令人意外，然而研究者或许会感兴趣的是，Y 与 X 的关系是否会有所改变，亦即较小的 X 以及较大的 X，其对 Y 的影响不一样，如同前一节所提到的，Morck, Shleifer and Vishny(1998) 发现：在不同区间的管理者持股状态，与公司价值的关系是不一样的，学术上以分段回归 (piecewise regression) 来进行这方面的研究。

所谓分段回归，是指在横截面的数据中，有一点存在使得该数据可以形成两条直线，此时数据可以适配如下：

$$y = (\alpha_1 + \beta_1 x)D_1 + (\alpha_2 + \beta_2 x)D_2 + \varepsilon$$

模型中， D_1 与 D_2 为虚拟变量，当数据为第一部分时 D_1 为 1，其余为 0，若数据为第二部分时， D_2 为 1，其他为 0，

而如果转折点为 γ ，该模型会有如下特性： $\alpha_1 + \beta_1 \gamma = \alpha_2 + \beta_2 \gamma$ ，移项后可得 $\alpha_2 = \alpha_1 + \beta_1 \gamma - \beta_2 \gamma$ ，将其代入回归模型中，可改写为如下：

$$y = (\alpha_1 + \beta_1 x)D_1 + (\alpha_1 + \beta_1 \gamma - \beta_2 \gamma + \beta_2 x)D_2 + \varepsilon$$

$$y = \alpha_1(D_1 + D_2) + \beta_1(xD_1 + \gamma D_2) + \beta_2(x - \gamma)D_2 + \varepsilon$$

$$y = \alpha_1 + \beta_1(xD_1 + \gamma D_2) + \beta_2(x - \gamma)D_2 + \varepsilon$$

$$y = \alpha' + \beta_1 x D_1 + \beta_2(x - \gamma)D_2 + \varepsilon \quad \alpha' = \alpha_1 + \beta_1 \gamma D_2$$

在估计完该模型之后，在 β_2 不为 0 的情况下，我们需进一步检定 Chow test 是否显著，如果 β_2 检定结果未达显著，显示数据两段的回归系数相同，则表示分段不存在；若分段回归系数显著，则仍须验证该两段数据有结构性改变的现象。

但是如同结构改变点一样，有时候并不能确定正确的分段点，或者邻近的数值有更佳的分段点，我们可能在连续的数点中，都能找到几个可以使两样本分段的点，并且通过 Chow test，又或者分段点可能不止一个，该如何进行这方面的研究呢？学者 Cho (1998) 提出了方格搜寻法来找出适合转折点的方法，假设在 x 中有四个转折点， h_1 、 h_2 、 h_3 、 h_4 ，

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \varepsilon$$

其中，当 $x \leq h_1$ 时， $x_1 = x$ ，其余为 0；当 $h_1 < x \leq h_2$ ， $x_2 = x - h_1$ ，其余为 0；当 $h_2 < x \leq h_3$ ， $x_3 = x - h_2$ ，其余为 0；当 $h_3 < x \leq h_4$ ， $x_4 = x - h_3$ ，其余为 0；当 $x > h_4$ 时， $x_5 = x - h_4$ 。

首先将 x 由小到大进行排序，进行下式：

$$y = \alpha + \beta_1 x_1 + \varepsilon$$

在该模型中，找出 h_1 能使得估计出来的 R^2 最大，此时佐以 Chow test，来分析结构改变是否存在，接着进行下式：

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

同样的，我们找出 h_2 能使得估计出来的 R^2 最大，且 β_2 为不为 0 的数值，此时佐以 Chow test，来分析结构改变是否存在，依序找出 h_3 、 h_4 。以下以 example_12_6.sas 为例来介绍如何进行分段回归。

产生 3 个不同时期数据的语法如表 12-26 所示。

表 12-26

```
data a;
do t=1 to 500;
x=rannor(1);
if x<=-0.5 then y=1-2*x+rannor(2);
else if -0.5<x=<0.5 then y=2+10*x+rannor(2);
else y=      1-6*x+rannor(2); /*产生有三段数据的数据*/
drop t;
output;
end;
run;
```

在此处仿真出来的数据中，理论上应该有三段数据，但是由于是随机生成随机数，真实数据中未必有 0 以及 0.5 这两个分段点，所以实际上还是需要依赖分析找出真正的分段点在何处。
检验第一个产生结构式改变的时点的语法如表 12-27 所示。

表 12-27

```
proc sort;by x;
run;
data a;
set a;
t=_n_;
run;
data a;
set a;
do i=10 to 490;
output;
end;
run;
data a;
set a;
if i>=t then d1=1;else d1=0;
if d1=1 then d2=0;else d2=1;
run;
proc sort data=a;by i d1;
run;
proc reg data=a noprint outest=k(keep=_p_ i d1 _rsq_) adjrsq;
model y=x;
by i d1;
```

续表

```
output out=a r=r2;  
quit;  
proc reg data=a noprint;  
model y=x;  
output out=a r=r1;  
by i;  
quit;
```

将数据依照 x 进行排序，为了避免数据太早出现分段，不适合也不需要情况发生，通常会先将前后几个%的数据先排除掉，接着产生两笔虚拟变量 $d1$ 以及 $d2$ ，计算做 Chow test 需要的数据，注意在研究逻辑上，先找出分段点再进行 Chow test，但是在程序进行中，可以同时计算，在找出分段点的同时顺便对数据进行 Chow test 检定，而由于第一步需要找出使 R^2 最大的 β_1 数据，在 $outest=k$ 中，此处要求输出 $rsq_$ 这个变量。

求得 R^2 最大的断点语法如表 12-28 所示。

表 12-28

```
proc sort data=k;by descending d1 descending rsq_;  
run;  
data k;  
set k;  
if _n_=1 then output;  
run;
```

要求 SAS 依照 $D1$ 以及 $rsq_$ 由大到小排序，这样一来第一笔观测值就是满足让 R^2 最大的点，接下来针对该点进行分段回归的系数估计。

检验 R^2 最大点是否产生结构改变的语法如表 12-29 所示。

表 12-29

```
data a;  
merge k a;by i;  
if _p_=, then delete;  
run;  
data b;  
set a;  
knot=x;  
if i=t then output;  
keep knot i;  
run;
```

续表

```

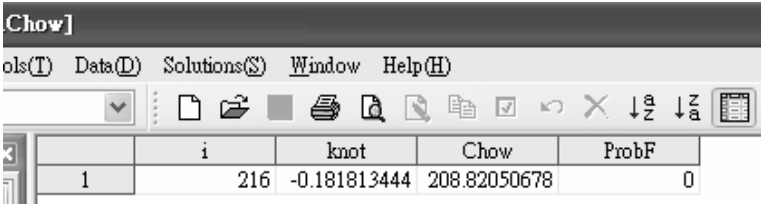
data a;
merge a b;by i;
du1=x*d1;
du2=(x-knot)*d2;
run;
proc sort data=a;by i;
run;
proc reg data=a noprint outest=b adjrsq;
model y=du1 du2;
by i ;
quit;
data a;
set a;
ESS1=r1**2;
ESS2=r2**2;
run;
proc means data=a noprint;
var ess1 ess2 knot;
by i;
output out=aa sum=ESS1 ESS2 knot n=n;
run;
proc means data=k noprint;
var _p_;
by i;
output out=k mean=_p_ n=k;
run;
data aa;
merge aa k;by i;
knot=knot/n;
Chow=((ESS1-ESS2)/_p_)/(ESS2/(N-k*_p_));
ProbF=1-probf(Chow,_p_,N-2*_p_);
keep i Chow ProbF knot;
run;
proc means noprint data=b;
var _adjrsq_;
by _DEPVAR_;
output out=c(drop=_type_ _freq_) max=r;

```

续表

```
run;
data b;
merge b c;by _DEPVAR_;
if _adjrsq_ =r then output;
keep i intercept du1-du100;
run;
data beta;
set b;
beta1=du1;
keep i beta1;
run;
data chow;
set aa;
run;
```

以上的部分分别使用 Chow test 检定,并且将分段点取出来,Chow 检验结果的数据如图 12-7 所示。



	i	knot	Chow	ProbF
1	216	-0.181813444	208.82050678	0

图 12-7

由文档可知分段点在-0.1818 这个点之上,但是仍不知道系数值是否有显著性,下面为参数的模型估计。检验断点系数值显著性的语句如表 12-30 所示。

表 12-30

```
proc reg data=a;
model y=du1 du2;
quit;
```

此处同时检定 du1 以及 du2 是否不为 0,显示第一个分段点是合适的,接下来找寻第 2 个分段点,在第 2 个分段点以后可以使用宏程序语法。多断点使用的宏语法如表 12-31 所示。

表 12-31

```
/*第二段以后可以使用的宏程序*/
%macro piecewise(du1,du2,nobs);
data a;
```

续表

```

merge a beta;by i;
start=i;
keep y x y t start du1-du100 ;
run;
data a;
set a;
do i=start+5 to &nobs-10;
output;
end;
run;
data a;
set a;
if i>=t then d1=1;else d1=0;
if d1=1 then d2=0;else d2=1;
run;
proc sort data=a;by i d1;
run;
proc reg data=a noprint outest=k(keep=_p_ i d1 _rsq_) adjrsq;
model y=du1- &du1;
by i d1;
output out=a r=r2;
quit;
proc reg data=a noprint;
model y=du1-&du1;
output out=a r=r1;
by i;
quit;
proc sort data=k;by descending d1 descending _rsq_;
run;
data k;
set k;
if _n_=1 then output;
run;
data a;
merge k a;by i;
if _p_=, then delete;
run;

```

```
data b;
set a;
knot=x;
if i=t then output;
keep knot i;
run;
proc sort data=a;by i;
run;
data a;
merge a b;by i;
&du1=&du1*d1;
&du2=(x-knot)*d2;
run;
proc sort data=a;by i;
run;
proc reg data=a noprint outest=b adjrsq;
model y=du1-&du2 ;
by i ;
quit;
data a;
set a;
ESS1=r1**2;
ESS2=r2**2;
run;
proc means data=a noprint;
var ess1 ess2 knot;
by i;
output out=aa sum=ESS1 ESS2 knot n=n;
run;
proc means data=k noprint;
var _p_;
by i;
output out=k mean=_p_ n=k;
run;
data aa;
merge aa k;by i;
knot=knot/n;
```

续表

```
Chow=((ESS1-ESS2)/_p_)/(ESS2/(N-k*_p_));
ProbF=1-probf(Chow,_p_,N-2*_p_);
keep i Chow ProbF knot;
run;
proc means noprint data=b;
var _adjrsq_;
by _DEPVAR_;
output out=c(drop=_type_ _freq_ ) max=r;
run;
data b;
merge b c;by _DEPVAR_;
if _adjrsq_=r then output;
keep i intercept du1-du100;
run;
data beta;
set b;
keep i ;
run;
data chow;
set chow aa;
run;
proc reg data=a;
model y=du1-&du2;
quit;
%mend;
```

在以上的宏程序中，仅需要了解 du1、du2 以及 nobs，本质上 du1 为下一个分段点要估计的参数值，du2 为第二个参数值，nobs 则为总样本数，先前的部分是针对 du1 以及 du2 估计，接下来则是要估计 du2 以及 du3，因此在执行程序上可以输入如表 12-32 所示的多断点使用范例的宏语法。

表 12-32

```
%piecewise(du2,du3,500);
%piecewise(du3,du4,500);
```

双断点检验的结果如图 12-8 所示，第二个分段点亦是结构改变点，该点为 0.48337。

	i	knot	Chow	ProbF
1	216	-0.181813444	208.82050678	0
2	359	0.4833790002	548.88705128	0

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	1	-0.61587	0.11143	-5.53	<.0001
du1	1	-2.95440	0.11613	-25.44	<.0001
du2	1	11.71460	0.40260	29.10	<.0001
du3	1	-7.18962	0.18290	-39.31	<.0001

图 12-8

虽然进行了第三个分段点的估计，但是结果显示不存在第三个结构改变的片段。第三个断点检验的结果如图 12-9 所示。

Chow					
ols(T) Data(D) Solutions(S) Window Help(H)					
	i	knot	Chow	ProbF	
1	216	-0.181813444	208.82050678	0	
2	359	0.4833790002	548.88705128	0	
3	490	1.9252967797	1.059864435	0.3757903299	

图 12-9

12.7 总结

本章介绍了回归语法的进阶语法，除了简单的横截面单期回归外，本章还介绍了跨期的回归语法，教大家移动窗口以及滚动式语法的程序撰写，更进一步，介绍了结构式改变语法以及其在分段回归上的应用。此处要特别说明的是，分段回归与门限回归虽然相似，但并不相同，分段回归单纯指的是特定 X 变量与 Y 变量的回归影响有所不同，在实证上并不放入其他控制变量，而门限回归则纳入其他变量，但不探讨分段的变量，且其检定方法须采用拔靴法，在本书的 17.2 节会详细介绍拔靴法。

第 13 章

panel data (proc panel、proc tscsreg)

在第 11 和第 12 章介绍了一些最小平方估计式的回归方法，然而在财务研究中，研究者常常需要使用跨期间的数据，由于有不同公司与不同期间，这些数据本身可能会存在异质变异性，因此在进行回归时，会设立虚拟变量以控制这些公司的个别效果（individual effect），这在研究中我们称之为公司固定效应（firm fixed effect），而若针对时间变量设立虚拟变量，则称之为时间固定效应（time fixed effect），而这些数值就成为了特定公司或者特定期间的截距项的效果。

除此之外，还有所谓的随机效应（random effect），即虽然公司或者不同期间有其特定的异质变异，但实际上无法确定这些效果的固定的截距项，因此会假设不同公司或者时间的平均截距项为 0，通常研究者无法确认 panel data 应该是使用固定效应估计较佳还是使用随机效应较佳，因此会使用 Hausman test 来检验数据应该以何种模型来估计。13.1 节介绍固定效应与随机效应的程序语法，在 13.2 节中介绍如何选择适合的模型，即进行 ols 估计或者是使用固定效应还是随机效应模型。最后则希望能提供一个标准的宏语法，可帮助读者自动挑选适合使用的估计模型。

13.1 固定效应与随机效应的估计方法

本节介绍固定效应的估计方法，在此之前介绍 balance panel data 以及 unbalance panel data，所谓 balance panel data 指的是样本公司的公司数在所有的样本期间内都存在才纳入研究，例如研究期间共 10 年，探讨 20 家公司，则总观测值为 200 个，如果有一家公司有遗漏值，则必须将之删除，而在财务研究中，这会产生样本选择性偏误（sample selection bias），因为有些公司可能会在样本期间的第五年下市，研究者便不采用，或者有些公司在样本期间的第八年上市，研究者也不采用其数据，投资者完全无法预测一家公司会在五年后下市，所以其进行投资决策或分析时，必然会考虑这家未来会下市的公司，而在进行财务研究时，基本上不应该忽略不计，而应运而生的就是 unbalance panel data，该数据即是指样本数据不需要在样本期间都存在才考虑，在估计上比较符合真实情况。在早期的软硬件不佳的情况下，使用 balance panel data 估计确实较为方便，但是现在的计算机硬件已有了大幅度的提高，足以进行快速的运算，而在 SAS 这两种数据都是采用相同的语法，亦即若是 unbalance data，估计出来就是 unbalance panel data 的模型。但要特别注意，如果是 unbalance panel data，在估计随机效应模型上会花较长的时间。以下以 example_13_1.sas 为例来介绍该语法。产生 Panel data 的公司与时间虚拟变量的语句如表 13-1 所示。

表 13-1

```
DM'LOG;CLEAR;OUT;CLEAR';
option nonotes nolabel;
libname aa 'D:\SAS 在财务研究上的运用\SAS data\月股价\';
data a;
set aa.price;
if nmiss(ret,pb,turn)^=0 then delete;
month=y*100+m;
if y=2006 and int(code/100)=11 then output;
keep code month ret pb turn;
run;
proc sort data=a out=month(keep=month) nodupkey;by month;
run;
proc sort data=a;by code;
run;
proc means noprint data=a;
var ret;
output out=b(keep=code n) n=n;
by code;
run;
data final;
```

续表

```
merge a b;by code;
if n^=12 then delete;
run;
proc sort data=final out=code(keep=code) nodupkey;by code;
run;
%macro dummy(file,name,n);
data &file;
set &file;
%do i=1 %to &n;
if _n_=&i then &name&i=1;else &name&i=0;
%end;
run;
%mend;
%dummy(month,month,12);
%dummy(code,code,7);
data final;
merge final code;by code;
run;
proc sort data=final;by month;
run;
data final;
merge final month;by month;
run;
```

为了避免虚拟变量过多，在此处仅对 2006 年的水泥行业进行估计，设定好时间虚拟变量以及公司虚拟变量后就进行回归模型。进行有公司与时间两种虚拟变量的回归语句如表 13-2 所示。

表 13-2

```
proc reg data=final;
model ret=turn pb;
model ret=turn pb code1-code7;
model ret=turn pb month1-month12;
model ret=turn pb code1-code7 month1-month12;
quit;
```

在此处我们进行 4 个模型的研究，模型 1 为不放入固定效应的模型，模型 2 为放入了公司固定效应的模型，模型 3 为放入了时间固定效应的模型，模型 4 则为将两种固定效应都放入了的模型，实际

上后面 3 个模型都有问题，在看完估计截距项结果以后再作说明。无虚拟变数的回归结果如图 13-1 所示。

Root MSE	6.77682	R-Square	0.2223
Dependent Mean	4.23810	Adj R-Sq	0.2031
Coeff Var	159.90238		

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	1	-1.29403	2.28026	-0.57	0.5719
turn	1	0.40930	0.09157	4.47	<.0001
PB	1	2.74209	2.49644	1.10	0.2753

图 13-1

由模型 1 的估计结果可以发现，周转率对报酬率有正向显著的解释能力，而其 R^2 为 0.2 左右，有问题的是模型 2、模型 3、模型 4。纳入公司虚拟变数回归结果如图 13-2 所示。

Root MSE	6.30133	R-Square	0.3774
Dependent Mean	4.23810	Adj R-Sq	0.3110
Coeff Var	148.68317		

NOTE: Model is not full rank. Least-squares solutions for the parameters are not unique. Some statistics will be misleading. reported DF of 0 or B means that the estimate is biased.
NOTE: The following parameters have been set to 0, since the variables are a linear combination of other variables as shown.

code7 = Intercept - code1 - code2 - code3 - code4 - code5 - code6

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	B	-9.48131	3.95407	-2.40	0.0190
turn	1	0.25278	0.11311	2.23	0.0284
PB	1	26.64693	7.48930	3.56	0.0007
code1	B	-24.84885	6.52741	-3.81	0.0003
code2	B	-17.11277	5.58145	-3.07	0.0030
code3	B	-14.23520	3.77751	-3.77	0.0003
code4	B	-4.68009	2.57908	-1.81	0.0736
code5	B	-9.10369	3.02643	-3.01	0.0036
code6	B	-7.99473	3.24850	-2.46	0.0162
code7	0	0	.	.	.

图 13-2

在模型 2 中， R^2 提高到 0.3774，turn 的系数值为 0.252、PB 的系数值 26.646，但出现了共线性问题，实际上，固定效应就是指不同公司的截距项，在进行 panel data 的固定效应的估计时，就不能放入所有公司的虚拟变量，因为模型本身已经有一个截距项了，7 家公司是不会有 8 个截距项的，相同的问题也产生在月份上，模型 3 仅能放 11 个时间虚拟变量，那么模型 4 能放入几个呢？我们来看看如图 13-3 所示的纳入公司与时间两个虚拟变量回归的结果。

结果仅能放入 $(7-1) + (12-1)$ 个虚拟变量，行文至此，读者可能会有所疑问，如果样本期间变长，公司家数变多，那么在表格上将如何报告呢？在学术研究上针对固定效应的回归，通常标记进行

公司固定效应或者时间固定效应，甚者有些文献则不报告截距项，因为截距项属于特定公司或期间的固定效应，报告其数值的意义不大。

```
code7 = Intercept - code1 - code2 - code3 - code4 - code5 - code6
month12 = Intercept - month1 - month2 - month3 - month4 - month5 - month6 - month7 - month8 - month9 - month10 - month11
```

Parameter Estimates					
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > t
Intercept	B	-21.52848	6.41307	-3.36	0.0013
turn	1	0.38797	0.09511	4.08	0.0001
PB	1	29.33695	8.58196	3.42	0.0011
code1	B	-26.92689	7.09842	-3.79	0.0003
code2	B	-18.32814	5.77111	-3.18	0.0023
code3	B	-16.11844	3.98167	-4.05	0.0001
code4	B	-4.99775	1.90098	-2.63	0.0107
code5	B	-9.88515	2.67977	-3.69	0.0005
code6	B	-8.07960	2.76777	-2.92	0.0048
code7	0	0	.	.	.
month1	B	8.93258	3.50807	2.55	0.0133
month2	B	10.73278	3.45166	3.13	0.0027
month3	B	10.24756	3.36275	3.05	0.0034
month4	B	17.22560	2.89391	5.95	<.0001
month5	B	5.24790	2.67982	1.96	0.0546
month6	B	3.93437	3.24260	1.21	0.2295
month7	B	8.59690	3.38725	2.54	0.0136
month8	B	10.58032	3.49004	3.03	0.0035
month9	B	14.99358	3.08326	4.86	<.0001
month10	B	12.58407	2.73354	4.60	<.0001
month11	B	12.00850	2.46412	4.87	<.0001
month12	0	0	.	.	.

图 13-3

虽然 panel data 可以使用 proc reg 来估计，但却需要因为不同公司而放入其他的虚拟变量，为此 SAS 提供了专门为 Panel data 的程序语法：proc panel 与 proc tscsreg 的语法，就功能而言 proc panel 的语法较为完整，但是该语法是在 SAS 9.2 版本以后才纳入的，SAS 9.1 的版本仅能使用 proc tscsreg。

使用如表 13-3 所示的采用 Panel 程序步进行固定效应回归的方法，研究者能估计出 panel data 的公司固定效应，由于 panel data 同时有时间与横截面的数据，因此在进行模型估计前必须要先排序。在只考虑单一固定效应的时候，如果要看公司固定效应，则应先对公司变量排序，而在 id 之后一定要摆放两个变量，一个指定公司，另一个指定时间，此处是先针对公司进行估计，故将公司摆在前面，如果是先估计时间，就是将时间变量放在前面。采用 Panel 程序步进行回归的结果如图 13-4 所示。

表 13-3

```
proc sort data=a;by code month;
run;
proc panel data=a;
model ret=turn pb/fixone;
id code month;
run;
```

Fit Statistics						
	SSE	2978.0113	DFE	75		
	MSE	39.7068	Root MSE	6.3013		
	R-Square	0.8774				
Parameter Estimates						
Variable	DF	Estimate	Standard Error	t Value	Pr > t	Label
CS1	1	-24.8488	6.5274	-3.81	0.0003	Cross Sectional Effect 1
CS2	1	-17.1128	5.5814	-3.07	0.0030	Cross Sectional Effect 2
CS3	1	-14.2352	3.7775	-3.77	0.0003	Cross Sectional Effect 3
CS4	1	-4.68009	2.5791	-1.81	0.0736	Cross Sectional Effect 4
CS5	1	-9.10369	3.0264	-3.01	0.0036	Cross Sectional Effect 5
CS6	1	-7.99473	3.2485	-2.46	0.0162	Cross Sectional Effect 6
Intercept	1	-9.48131	3.9541	-2.40	0.0190	Intercept
turn	1	0.252779	0.1131	2.23	0.0284	
PB	1	26.64693	7.4893	3.56	0.0007	

图 13-4

就 R^2 与系数值的估计来看, 使用 proc panel 估计跟采用 proc reg 估计是一模一样的, 而如果要同时估计两种固定效应, 则只要将 fixone 改成 fixtwo 即可, 同时考虑公司与时间固定效应的语句如表 13-4 所示。

表 13-4

proc panel data=a;
model ret=turn pb/fixtwo;
id code month;
run;

接下来, 介绍 proc tscsreg 的语法, 实际上在估计一般的 panel data 模型时, 这两个语法是一模一样的, 采用 tscsreg 程序步回归的语句如表 13-5 所示。

表 13-5

/*由于 proc panel 是在 SAS 9.2 版以后才出现的
sas9.1 以前的版本 可以使用 proc tscsreg 的语法*/
proc tscsreg data=a;
model ret=turn pb/fixone;
id code month;
run;
proc tscsreg data=a;
model ret=turn pb/fixtwo;
id code month;
run;

虽然基础的 panel data 两种语法都能估计，但是如果读者需要用到进阶语法，例如 dynamtic Panel data 时，就需要使用 proc panel 来进行估计。

在本节最后，我们介绍如何估计随机效应模型。采用 panel 程序步进行随机效应的语句如表 13-6 所示。

表 13-6

<pre>proc panel data=a ; model ret=turn pb/ranone; id code month; run; proc panel data=a ; model ret=turn pb/rantwo; id code month; run;</pre>

行文至此，读者会发现 SAS 在撰写 proc panel 时非常方便，估计固定效应或者随机效应，只要撰写 fixone、fixtwo、ranone 或者 rantwo 就可以求得所需要的模型估计，接下来，我们探讨如何挑选适合的模型。

13.2 panel data 的实证流程

13.1 节使用了 balance panel data 来简单介绍如何估计固定效应与随机效应模型，然而在实证上应该采用这样的步骤来进行：第一步，检验该数据是否适合使用 Panel data 来进行估计，如果不适合，就使用 ols，若适合则进行下一步骤；第二步是检验该 Panel data 适合使用固定效应模型或者是随机效应模型，如果是适合随机效应模型，就使用 ranone 与 rantwo，若适合使用固定效应模型，就使用 fixone 与 fixtwo；第三步，如果使用固定效应模型，仍须进行联合检定，检定所有的固定效应是否相等，如果固定效应的系数值都相等，其实也就等于固定效应不存在。以下以 example_13_2.sas 为例来介绍面板数据的检定以及估计。进行 Panel data 回归的数据预处理的语法如表 13-7 所示。

表 13-7

<pre>DM'LOG;CLEAR;OUT;CLEAR'; option notes nlabel; libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price'; data a; set aa.price; if nmiss(ret,pb,turn)^=0 then delete; month=y*100+m;</pre>

续表

```
if y=2006 then output;
keep code month ret pb turn;
run;
proc sort data=a;by code;
run;
proc means noprint data=a;
var ret;by code;
output out=code n=n;
run;
data a;
merge a code;by code;
run;
```

此处采用前一节的方法，一样将数据做一点计数的处理，虽然在本节的运用中都要使用 unbalance panel data 来进行介绍，进行 unbalance panel data 虽然不要求样本公司在整个样本期间都存在，但是仍然要求样本公司至少要有 2 个观测值才能进行模型估计，要求数据至少要有 2 个观测值的语句如表 13-8 所示。

表 13-8

```
proc panel data=a (where=(n>1));
model ret=turn pb/bp bp2;
id code month;
run;
```

此处在使用数据估计时，在数据源的地方加入条件(where=(n>1))，即告知 SAS 只要针对观测值大于 1 的公司进行估计就可以了，需要注意的是变量 *n* 是样本公司的观测值，是由于先前的部分计算所得的，并非是 SAS 内建的语法。该语法执行后的 Panel data 的关键统计检定结果如图 13-5 所示。

Hausman Test for Random Effects			
DF	m Value	Pr > m	检验固定效应和随机效应模型
2	213.56	<.0001	显著应采用固定效应模型
Breusch Pagan Test for Random Effects (One Way)			
DF	m Value	Pr > m	
1	28.59	<.0001	检验 Panel1 和 OLS
Breusch Pagan Test for Random Effects (Two Way)			
DF	m Value	Pr > m	显著应采用 Panel1
2	183.41	<.0001	

图 13-5

在 model ret=turn pb/bp bp2; 中，在斜线后输入 bp bp2 是为了检定应该使用 ols 估计或者是使用随机效应模型估计，所谓 bp bp2 就是由 Breusch and Pagan (1980) 提出的 LM test¹，主要是针对检定应该使用 ols 模型或者 Panel data 的随机效应模型，若 LM test 的分数不显著，则使用 ols 模型，若显著，则需要进一步检验应该使用固定效应模型或者随机效应模型，如果不指定固定效应或随机效应模型，SAS 会默认以随机效应模型先行估计。

检验固定效应或者随机效应模型是采用 Hausman (1978) 提出的 Hausman 检定，在 SAS 的 proc panel 的程序中，Hauman 检定会自动产生并计算为 M 分数，该数值如果显著，就应该使用固定效应模型，若不显著，则采用随机效应模型估计即可。所以就该数据而言，应该采用固定效应模型来进行分析，非平衡面板数据的语法如表 13-9 所示。

表 13-9

proc panel data=a (where=(n>1));			
model ret=turn pb/fixone;			
id code month;			
run;			
proc sort data=a;by month code;			
run;			
proc panel data=a (where=(n>1));			
model ret=turn pb/fixone fixtwo;			
id month code;			
run;			

由表 13-9 可见，求出在公司固定效应模型下的系数估计，接着依照时间、公司的顺序排序，同时进行时间固定效应以及时间和公司固定效应两个模型一起的系数估计，在 SAS 中会同时进行固定效应是否存在的联合检定（F 分数）。检定固定效应与时间效应是否存在的情况如图 13-6 所示。

F Test for No Fixed Effects			
Num DF	Den DF	F Value	Pr > F
748	7923	2.33	<.0001

F Test for No Fixed Effects			
Num DF	Den DF	F Value	Pr > F
11	8660	101.21	<.0001

F Test for No Fixed Effects			
Num DF	Den DF	F Value	Pr > F
759	7912	3.74	<.0001

图 13-6

1 要注意的是，proc tscsreg 并未提供 LM 检定的语法。

由图 13-6 的结果可知，公司固定效应存在、时间固定效应存在，而且同时考虑公司与时间固定效应也得到固定效应存在的结果，接下来就可以检验在固定效应模型下的系数值了。简单来说，如果要完整进行 panel data 的流程，要撰写两次 Panel data 的语法，第一次撰写 bp bp2，第二次估计固定效应即可；但若是第一次检定结果显示使用 ols 即可，则可直接采用 proc reg 来进行模型估计。

接下来，依照进行 Panel data 的逻辑撰写一个标准化的宏，读者在需要进行 Panel data 时，只需要导入宏语法就可以顺利估计需要的结果了。

13.3 格式化 panel data 模型输出

本节介绍作者撰写的 panel 宏语法的使用方法，由于相关宏语法过于繁复，故本文不加赘述，仅以 example_13_3.sas 为例来介绍如何使用该宏语法，面板数据预处理的语法如表 13-20 所示。

表 13-20

```
libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price';
data a;
set aa.price;
if nmiss(ret,pb,turn,volum)^=0 then delete;
month=y*100+m;
if y=2006 then output;
keep code month ret pb turn volum;
run;
proc sort data=a;by code;
run;
proc means noprint data=a;
var ret;by code;
output out=code n=n;
run;
data a;
merge a code;by code;
run;
```

在此处，仍然采用 13.2 节的数据处理方式，只是我们额外增加了 volum 的变量，要特别注意的是进行 panel data 的模型不能有缺失值，而且必须先行计算每只股票的总存在期间，在后续的程序中，采用的变量为 *n*，若读者希望更改，则可自行修改宏程序，但必须将进行 panel data 估计的文档存储为 *a* 方能进行。面板数据宏语法的使用范例如表 13-21 所示。

表 13-21

```
%include 'D:\THE APPLICATION OF SAS IN FINANCIAL RESEARCH\CH13\program_cn\Panel.sas';  
%panel(a,ret,pb turn,code,month,2);
```

在此处,先要求 SAS 读取 panel.sas 的程序,该程序为 panel 的宏,接下来撰写%panel(ret, pb turn, code, month, 2), 其中 ret 为被解释变量, pb turn 为解释变量, 两者合一即要求 SAS 估计下列模型:

$$ret = \alpha + \beta_1 PB + \beta_2 Turn + \varepsilon$$

而 code 则为公司效果的变量, 如以美国的数据而言, 则可能是 cusip 这样的变量, month 则是时间效果的变量, 如果读者的数据是以 firm 来区分不同公司的, 以 t 来区分不同的时间, 则可以输入 firm, t 来表示, 最后一个语句中的数字 2 表示系数估计是要进位到小数点后的第几位。面板数据结果输出的语句如表 13-22 所示。

表 13-22

```
proc export data=panel_data  
outfile="D:\The Application of SAS in Financial Research\CH13\EXCELoutput\Panel"  
dbms=xlsx  
replace;  
sheet="model1";  
run;
```

在宏执行完毕后, SAS 会自动产生一个文档名为 panel_data 的 table, 可以将它输出到外部的 Excel 文档, 由于宏一次只能针对一个模型进行估计, 所以如果读者要估计多个模型, 就需要指定不同的文档, 或者如本例所示, 自行指定工作表名称, 在此命名为 model1。接下来可以再进行第二个模型。面板数据的第二个模型分析与输出的语句如表 13-23 所示。

表 13-23

```
%panel(a,ret,pb turn volum,code,month,3);  
proc export data=panel_data  
outfile="D:\The Application of SAS in Financial Research\CH13\EXCELoutput\Panel"  
dbms=xlsx  
replace;  
sheet="model2";  
run;
```

此处, 估计的是 $ret = \alpha + \beta_1 PB + \beta_2 Turn + \beta_3 volum + \varepsilon$, 并且要求系数值准确到小数点后的第 3 位, 并且将产出的结果输出为 model2 的工作表, 面板数据的 Excel 输出结果如图 13-7 所示。

varname	FixOne	FixTwo	Fixtime	Pooled_OLS	RanOne	RanTwo			
Intercept	-5.42 (-1.35)	1.36 (0.35)	6.37*** (11.31)	-0.5** (-2.1)	-4.46*** (-9.6)	-2.67** (-1.98)			
PB	4.45*** (14.53)	3.38*** (11.46)	0.22** (2.24)	0.35*** (3.3)	1.76*** (9.19)	1.11*** (6.51)			
turn	0.31*** (28.18)	0.28*** (25.6)	0.15*** (20.87)	0.18*** (24.05)	0.28*** (29.24)	0.24*** (25.36)			
firm	YES	YES	NO	NO	YES	YES	若有控制公司效应, 则 firm 输入为 YES, 否则为 NO		
time	NO	YES	YES	NO	NO	YES	若有控制时间效应, 则 time 输入为 YES, 否则为 NO		
rsq	23.98%	31.73%	17.83%	7.27%	11.94%	8.56%	自动输出 LM 以及 Hausman,		
LM	38241.03***	38241.03***	38241.03***	38241.03***	38241.03***	38241.03***	LM 不显著, 则 select 输出 OLS, 建议采用普通最小二乘法		
Hausman	213.56***	213.56***	213.56***	255.75***	213.56***	213.56***	若 LM 以及 Hausman 皆显著, 输出 fix, 建议采用固定效应模型		
select	Fix	Fix	Fix	Fix	Fix	Fix	若 LM 显著而 Hausman 不显著, 输出 ran, 建议采用随机效应模型		

图 13-7

在输出的结果中, 依序为 FixOne、FixTwo、Fixtime、Pooled_OLS、RanOne 以及 RanTwo 共 6 个模型, 其中 FixOne 表示控制公司固定效应的单一效应模型、FixTwo 表示两个因子皆有控制的双固定效应模型, Fixtime 表示控制时间固定效应的单一效果模型, Pooled_ols 为简单的 ols 模型, 而在随机效应模型中, 由于只控制时间效应的随机模型较少, 所以没有 Rantime 这个模型, 其中 RanOne 表示为控制公司效应的单一效果模型, RanTwo 则为同时控制公司与时间效应的随机模型。

而在 13.2 节的介绍中, 说明要检验的数据是否可以使用 Panel data 以及应该采用随机效应模型或者固定效应模型的检定, 我们同时输出 Breusch and Pagan (1980) 提出的 LM 以及 Hausman (1978) 提出的 Hausman 分数, 并且依据 5% 的显著水平作为判断准则, 如果 LM 分数仅有 1 颗星的显著水平, 则 select 会输出 ols 的字样, 表示推荐使用最小平方估计来估计; 而在 LM 分数有 2 颗星以上的显著水平时, Hausman 亦达到 2 颗星的显著水平, 则会输出 Fix, 表示推荐使用固定效应模型的 Panel data; 最后则是推荐使用随机效应模型的 Ran。

本节介绍的 panel 宏虽然是针对 panel data 来处理的, 但事实上在一般的研究中也常需要进行公司效果与时间效果的控制, 在前面的章节中亦提到, panel data 的固定效应模型其实可以简单地使用 ols 估计即可, 但是设定虚拟变量又过于麻烦, 如果读者希望快速估计出固定效应的模型, 亦可采用宏语法进行估计。最后要提醒读者的是本节最后的宏仅提供 SAS 9.2 以上的版本使用, 若是其他的版本, 由于并没有 proc panel 的程序, 所以是无法求得估计的。

最后提供该宏语法的使用说明, 如表 13-24 所示。使用该宏语法, 需要指定来源文档、被解释变量、解释变量串 (亦即可多个变量)、公司变量以及时间变量 (皆只能一个), 以及系数要四舍五入到小数点后的几位。

表 13-24

/*%panel(file,y,x,firm,tim,bit);	
file	请输入要进行分析的文档
y	输入被解释变量
x	输入解释变量"串"
firm	输入公司变量
time	输入时间变量
bit	输入系数的最大位数
*/	

13.4 总结

本章介绍了 panel data 的估计方法，同时探讨了固定效应以及随机效应模型，并且提供了一个宏，可以快速估计不同模型及其需要的数据。然而，panel data 还有一些其他的计量处理方式，虽然本章并未探讨，读者仍可参考 SAS 的 Help 去搜寻 proc panel 来研究如何撰写。

第 14 章

罗吉斯特模型

先前的章节介绍了回归方法，其应变量也就是左手边被解释变量的连续性变量 (continue variable)，可以顺利地使用 ols 来进行分析，但是有时候会遇到非连续性的数据，例如，公司发生财务危机或公司没有发生财务危机，仅能观察到有或无的结果，在数据上就无法表示为 0.1 破产、或者-0.1 破产，仅能使用虚拟变量来为之分类，1 为破产，其他为 0，一般常见是使用 logit 以及 probit model 来进行分析。而除了数据本身是属于 0 或 1 的形态外，还有可能遭遇到某些数据有缺失、截断的现象，此时则可以采用 Tobit model 来进行实证分析，这三种方法由于左手边的应变量是有限制的，故又称为限制性应变量 (limited dependent variable)。

本章简单地介绍罗吉斯特模型，在 SAS 中是使用 proc logistic 来撰写的，14.1 节先简单介绍基础的 logit model，14.2 节介绍 conditional logistic regression 程序的撰写，14.3 节介绍 multinomial logistic regression 的语法，这三种模型在财务研究中的应用是很普遍的，有志于进行财务研究的人应该好好了解，而在 14.4 节中，介绍如何使用 logistic regression 来将数据进行分组，以及将事后的虚拟变量数据转换成事前的连续变量数据的方法。

14.1 logit model (logistic regression)

本节简单介绍如何使用 SAS 来估计 logit model，在 SAS 的 help 中对 logit model 有精彩的理论介绍，其重要性可见一斑。

就一个结果为有或无的数据来看，可以将 Y 编码为 0 或者 1 这两个可能的结果，并且假设 X 为其解释变量， $\pi = \Pr(Y=1|X)$ 为模型中的反应概率，那么其 logit model 可以如下：

$$\text{logit}(\pi) \equiv \log(\pi/(1-\pi)) = \alpha + \beta x$$

根据上式，如果 β 估计出来为 0.2，其结果是表示 x 变量的数值越高，发生结果 1 的概率就越大，然而系数值，是无法解释会增加概率的，我们需要将上式转换才行。

$$\begin{aligned}\pi/(1-\pi) &= e^{\alpha+\beta x} \\ \pi &= e^{\alpha+\beta x} - \pi(e^{\alpha+\beta x}) \\ \pi(1+e^{\alpha+\beta x}) &= e^{\alpha+\beta x} \\ \pi &= e^{\alpha+\beta x} / (1+e^{\alpha+\beta x})\end{aligned}$$

通常在解释方法上可以先将所有的 x 变量代入其平均值，求得基础的概率，接着分别计算各个解释变量增加 1 单位对概率值产生的影响，此时的解释为在离开平均值 1 单位对概率的影响，如果使用中位数，则解释变量在中位数增加 1 单位对概率值的影响不同于在均值的位置增加 1 单位，若为了行文方便，则不一定要计算其对概率的影响，根据系数的符号，若为正，则该数值增加会增加概率；如系数值符号为负，则该数值增加会降低发生的概率，以下采用 example_14_1.sas 来介绍，观察资产负债表的一些数据是否能预测公司会宣布减资，注意此处只检验是否会宣布减资，而非检验其是否进行减资。发生事件与未发生事件的处理如表 14-1 所示。

表 14-1

<pre>DMLOG;CLEAR;OUT;CLEAR'; option nonotes nolabel; libname aa 'D:\The Application of SAS in Financial Research\SAS data\event date'; data a; set aa.event; event=1; /*要与未发生事件的样本区分*/ y=y-1; /*因为要以前一年的数据来预测未来是否会减资，所以在处理上先减 1*/ run; proc sort data=a nodupkey;by code y; run;</pre>

由表 14-1 所知，先读取事件样本，并新设一事件变量为 1，且将年份都减去 1，最后，利用排序语法删除掉重复的数据，这是因为公司很有可能会在同一年两次宣布要减资。

下面介绍如表 14-2 所示的财务变量处理语句。

表 14-2

libname aa 'D:\The Application of SAS in Financial Research\SAS data\financial';					
data b;					
set aa.financial;					
if y<1990 then delete;					
run;					
proc sort data=b;by code y;					
run;					
data a;					
merge a b;by code y;					
if event=. then event=0;					
d1=liquidasset/liquiddebt;					
d2=longinvestment/totalasset;					
d3=longdebt/totalasset;					
if d3>1 or d1=. or d2=. or d3=. then delete;					
keep event d1-d3;					
run;					

在此部分，将数据做个整理，并且选定三个变量，第 1 个变量为流动资产对流动负债比，第 2 个变量为投资金额对总资产比，第 3 个变量为长期负债对总资产比，选定原因为减资基本上是公司要将钱还回股东，实质上会降低公司的流动资产，因此特别选用跟流动性有关的变量来做模型的估计。

在 SAS 中进行 logit model 的程序为 proc logistic，logistic 回归语句如表 14-3 所示。之后的语法与 ols 的 proc reg 相似，logistic 回归估计的结果如图 14-1 所示。

表 14-3

proc logistic data=a ;					
model event=d1-d3;					
quit;					

Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	4.5582	0.1559	854.6949	<.0001
d1	1	0.2277	0.0638	12.7222	0.0004
d2	1	-1.6598	0.3462	22.9821	<.0001
d3	1	-1.4702	0.5676	6.7084	0.0096

图 14-1

从图 14-1 所示的结果中，发现 3 个比率的系数值皆显著，其中统计量为 Chi-Square 分数，有别于常用的 t 统计量，此处有两个问题需要讨论，第一个是统计量问题，采用 Chi-Square 分数来判定显著水平，较为少见，虽然我们有 p-value 可以作为辅助，但是在表格呈现中，是否就无法采用 t 分数来报告呢？我们要了解 Chi-Square 分数有一个特性，即是当自由度等于 1 时，Chi-Square 会等于 Z^2 ，而 Z 分数等于当自由度为无穷大时的 t 分数，因此在表格的呈现上，可以使用 t 分数或者 Z 分数这两个较为熟悉的数值。第 2 个问题来自于 SAS 的默认值，在使用 proc sort 的时候，SAS 会将数据由小到大排序，而在此处进行 logit model 的时候，SAS 计算的概率是 1 或者是 0 呢？如图 14-2 所示的样本分布表的结果能给予解答。

Response Profile		
Ordered Value	event	Total Frequency
1	0	22321
2	1	225

Probability modeled is event=0.

图 14-2

该结果显示，SAS 系统根据 0（较小的数值）来计算概率，这个预设情况和目前常用的统计软件估计的默认状况不同，在进行研究分析上需要注意，SAS 的默认系统都是由小到大的，可以采用和 proc sort 相同的方法来进行处理，采用 desc 语法进行 logistic 回归的语法如表 14-4 所示。

表 14-4

<pre>proc logistic data=a desc; model event=d1-d3 /maxiter=100; quit;</pre>		
---	--	--

在语法中，加入了 desc（descending 的简写指令），就可以要求 SAS 计算较大数字的概率，而并不是针对 event=0 来进行分析，而此处也加了重要的选项，maxiter=100 是更改 SAS 反复计算到收敛的次數，其内建次数为 25 次，往往会有无法收敛的情况。

采用 desc 进行 logistic 回归的结果如图 14-3 所示。

Response Profile		
Ordered Value	event	Total Frequency
1	1	225
2	0	22321

Probability modeled is event=1.

Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-4.5582	0.1559	854.6949	<.0001
d1	1	-0.2277	0.0638	12.7222	0.0004
d2	1	1.6598	0.3462	22.9821	<.0001
d3	1	1.4702	0.5676	6.7084	0.0096

图 14-3

由图 14-3 可见在加入了 desc 该语法以后，模型概率的估计确实是以 1 为主的，然而，系数值已经完全变为负数，显示出数值越高，该事件发生的可能性越低，据此有两个可能的解释：第一，公司的流动资产以及营收越高，可能隐含该家公司正在成长，因此不会进行减资的动作；第二，从我们的数据来看，进行减资宣布的公司只有 225 家，而未进行的有 22321 家公司，数字过于悬殊，导致估计失真，应该进一步进行公司配对的情况，在实证研究上，通常采用在相同的产业中，其公司规模以及账面市值比相似的公司来进行配对，如此，更能帮助确定这 3 个比率的正确预测方向。

虽然，proc logistic 的语法与 proc reg 相似，但是仍有不同之处，例如在 proc reg 中可以同时估计不同的模型，然而在 proc logistic 中仅能估计一个模型。

纳入多模型后的 logistic 回归结果如图 14-4 所示，only one model 可以在 proc logistic 使用，如果想要估计多个模型就必须分开撰写，原因在于 logit model 是需要反复计算而求得收敛数据的，如果同时估计不同的模型，计算机资源可能不足。接下来介绍如何将 logit model 的结果输出到 Excel 中，以便快速地进行分析，利用 ods 输出 logistic 回归结果的语句如表 14-5 所示。

```
893 proc logistic data=a desc ;
894 model event=d1-d3 /maxiter=100;
895 model event=d1-d2 /maxiter=100;
896 quit;
ERROR: Only one MODEL statement can be used with each invocation of PROC LOGISTIC.
```

图 14-4

表 14-5

proc logistic data=a desc ;					
model event=d1-d3 /maxiter=100 rsquare;					
ods output ParameterEstimates=esti rsquare=rsq nobs=n;					
quit;					

由表 14-5 可见，一样使用 ods ouptut 可以输出必要的数 据，分别为参数估计、R² 以及观测值数目，为了输出 R² 的数据，在模型之后需要再加入 rsquare 这个选项。logistic 回归的 ods 输出结果如图 14-5 所示。

	Variable	DF	Estimate	StdErr	WaldChiSq	ProbChiSq	_ESTTYPE_
1	Intercept	1	-4.5582	0.1559	854.6949	< .0001	MLE
2	d1	1	-0.2277	0.0638	12.7222	0.0004	MLE
3	d2	1	1.6598	0.3462	22.9821	< .0001	MLE
4	d3	1	1.4702	0.5676	6.7084	0.0096	MLE

Label1	cValue1	nValue1	Label2	cValue2	nValue2
R-Square	0.0025	0.002450	Max-rescaled R-Square	0.0232	0.023160

	Label	N	NObsRead	NObsUsed	SumFreqsRead	SumFreqsUsed
1	Number of Observations Read	22546	22546	22546	22546	22546
2	Number of Observations Used	22546	22546	22546	22546	22546

图 14-5

接下来针对这三个文档进行处理，目的是与前面的章节一样，将数据整理成与一般实际论文一样的表格。logistic 的回归系数值整理的语句如表 14-6 所示。

表 14-6

<pre>data esti; set esti; retain t 0; t=t+1; if probchisq<0.01 then sig=round(estimate,0.01) '***'; else if 0.01<probchisq=<0.05 then sig=round(estimate,0.01) '**'; else if 0.05<probchisq=<0.1 then sig=round(estimate,0.01) '*'; else sig=round(estimate,0.01) ''; tv='(' left (round(estimate/stderr,0.01) ')'; tv2=(estimate/stderr)**2; run;</pre>											
---	--	--	--	--	--	--	--	--	--	--	--

首先针对参数值做分析，根据显著水平标上星号，如果其 p 值小于 1%水平，就是三颗星，介于 1%~5%就标上 2 颗星，如果是介于 5%~10%就标上 1 颗星，t 值则为参数除以标准误差便可得到，为了能够进行验算，特别将该数值计算平方，以便跟 Wald Chi-Square 做比较。Logistic 回归系数值的整理结果如图 14-6 所示。

	Variable	DF	Estimate	StdErr	WaldChiSq	ProbChiSq	_ESTTYPE_	t	sig	tv	tv2
1	Intercept	1	-4.5582	0.1559	854.6949	<.0001	MLE	1	~4.56***	(-29.24)	854.69490636
2	d1	1	-0.2277	0.0636	12.7222	0.0004	MLE	2	~0.23***	(-3.57)	12.722159449
3	d2	1	1.6598	0.3462	22.9821	<.0001	MLE	3	1.66***	(4.79)	22.982148486
4	d3	1	1.4702	0.5676	6.7084	0.0096	MLE	4	1.47***	(2.59)	6.7084043908

图 14-6

由结果可知，前面的推论是正确的，可以在表格中以 t 分数或者 Z 分数来呈现，这样的数据为一般人所熟知，可以轻易地由数值推算其是否显著。

logistic 的观测值与 R² 整理语句如表 14-7 所示，由此整理语句可分别整理参数值、t 值、R² 以及观测值，最后再将其合并，logistic 回归的最终结果如图 14-7 所示，可以将其整理到 Excel 文档中，在研究报告上可以方便地呈现。

表 14-7

<pre>data a1; set esti; model=left(sig); type=1; keep model t type variable; run; data a2;</pre>											
--	--	--	--	--	--	--	--	--	--	--	--

```

set esti;
model=tv;
type=2;
keep model t type ;
run;
data rsq;
set rsq;
model=left(round(nvalue1*100,0.01)||"%");
t=100;
type=1;
variable='R2';
keep model t type variable;
run;
data n;
set n;
model=left(NObsUsed||");
t=1000;
type=1;
variable='N';
if _n_=2 then output;
keep model t type variable;
run;
data final;
set a1 a2 rsq n;
run;
proc sort data=final out=final(keep=model variable);by t type;
run;

```

Variable	model
Intercept	-4.56*** (-29.24)
d1	-0.23*** (-3.57)
d2	1.66*** (4.79)
d3	1.47*** (2.59)
R2	0.25%
N	22546

图 14-7

最后将该部分语法整理为宏，并存储为文档，使用 logistic 宏语法的使用范例如表 14-8 所示。

表 14-8

<pre>%inc "D:\The Application of SAS in Financial Research\CH14\program_cn\logit.sas"; %macro model; event=d1 d2 d3 %mend; %logit(a,y,4,logit1); %macro model; event=d1 %mend; %logit(a,y,4,logit2); %macro model; event=d2 %mend; %logit(a,y,4,logit3); %macro model; event=d3 %mend; %logit(a,y,4,logit4); %macro model; event=d1 d2 %mend; %logit(a,y,4,logit5); %macro model; event=d1 d3 %mend; %logit(a,y,4,logit6); %macro model; event=d1 d2 d3 %mend; %logit(a,y,4,logit6); %logit_file(y,7,logit_result);</pre>
--

在本例中采用了分组进行罗吉斯特回归，y 为分组变量，并且进行了 7 个不同罗吉斯特模型，依然在第一个模型要写入所有的解释变量，而在执行%logit_file 这个宏语法后，就会将结果整理到 logit_result 中。

14.2 conditional logistic regression

在第 13 章中，我们采用所有公司以及发布增资的事件来介绍 logit mdoel 的程序语法，同时也提到当未发生事件的公司与发生事件的公司数目过于悬殊时，可能会使得结果不稳定，本节采用 conditional logistic regression 来分析三种财务比率对公司是否会发布增资的影响。

使用 conditional logistic regression 有一个先决条件，亦即发生事件的公司与未发生事件的公司是成对的（一对一、一对二甚至一对三），其配对样本表格数据如表 14-9 所示。

表 14-9

sn	code	event
1	1101	1
1	1102	0
1	1107	0
2	2303	0
2	2330	1
2	2352	0

该表格呈现了两家事件公司，分别对该事件 sn 编为 1 与 2，并且对这两家事件公司找寻 2 家产业相同的配对公司，如此一来事件公司与配对公司的比率就为 1 比 2，不会有两百家公司比一万多家公司的情况，反而使得想探讨的事件成为稀少性的公司，而由于公司样本是采用针对每一家事件公司进行配对的，而非单纯的两群样本，因此要使用 conditional logistic regression 来进行分析，以下以 example_14_2.sas 为例来介绍该语法。将数据区分为发生事件与未发生事件样本的语句如表 14-10 所示。

表 14-10

DM'LOG;CLEAR;OUT;CLEAR'; option nonotes nlabel; libname aa 'D:\The Application of SAS in Financial Research\SAS data\event date'; data a; set aa.event; event=1; /*要与未发生事件的样本区分*/ y=y-1; /*因为要以以前一年的数据预测未来是否会减资，所以在处理上先减 1*/ run ; proc sort data=a nodupkey;by code y;
--

续表

```
run;
libname aa 'D:\The Application of SAS in Financial Research\SAS data\financial';
data b;
set aa.financial;
if y<1990 then delete;
run;
proc sort data=b;by code y;
run;
```

该程序前面的部分与前一节的程序语法一样，重点在于整理所需要的财务数据，接下来采用较巧妙的方式来寻找配对样本。随机抽取配对样本的语句如表 14-11 所示。

表 14-11

```
data a1 a2;
merge a b;by code y;
if event=. then event=0;
d1=liquidasset/liqiddebt;
d2=longinvestment/totalasset;
d3=longdebt/totalasset;
if d3>1 or d1=. or d2=. or d3=. then delete;
keep event d1-d3;
if event=1 then output a1 ;else output a2;
run;
proc surveyselect data=a2 out=a2 noprint
method=srs seed=1 n=225;
run;
data a1;
set a1;
retain id 0;
id=id+1;
run;
data a2;
set a2;
retain id 0;
id=id+1;
run;
data a;
set a1 a2;
run;
```

在该程序中，将数据分为样本公司，以及未发生事件的公司，严谨的做法是进一步根据变量，对每家公司进行一对一的配对，此处为了程序介绍方便，我们仅任意抽出 225 家公司进行配对。并且将公司编码为 1 到 225。依照事件样本排序的语句如表 14-12 所示。样本排序的结果如图 14-8 所示。

表 14-12

```
proc sort data=a;by id;
run;
```

进行分析的数据形式如下，每一家事件公司会有一家相对应的未发生事件的公司，使用 conditional logistic regression 时，一定要将数据依照 id（或其他成对编号）做排序。样本排序的结果如图 14-8 所示。conditional logistic 回归语句如表 14-13 所示。

event	d1	d2	d3	id
1	0.9492720691	0.1571815234	0.2971854905	1
0	7.7154478185	0.0903898858	0.0009722974	1
1	0.1854215927	0.0987106003	0	2
0	0.9400926993	0.2430718858	0.1330816614	2
1	1.0898012642	0.0783965881	0.0240088322	3
0	1.6717611763	0.0171240217	0.0047525842	3
1	0.3250263005	0.3775130127	0.3307440525	4
0	1.4208219727	0	0	4
1	0.3058815698	0.4067754221	0.4138595469	5
0	0.6294750951	0	0.0497679589	5

图 14-8

表 14-13

```
proc logistic data=a desc;
model event=d1-d3/ maxiter=100;
strata id;
quit;
```

在如表 14-13 所示的程序中，新增了一个语法 strata id，使用该语法 SAS 就会自动针对 id 进行 logistic regression，conditional logistic 回归的结果如图 14-9 所示。

Analysis of Conditional Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
d1	1	-0.0663	0.0457	2.1034	0.1470
d2	1	2.8638	0.7751	13.6532	0.0002
d3	1	2.3251	1.1249	4.2721	0.0387

图 14-9

使用 strata id 之后，SAS 会呈现 conditional analysis，而且亦不会估计截距项的系数值，而根据 conditional logistic regression 的结果来看，似乎这三种财务比率对增资宣布的影响仍然是负的。接下来仍采用相同的方法将结果直接整理成可以方便报告的形式。conditional logistic 回归整理的输出语句如表 14-14 所示。

表 14-14

```

proc logistic data=a desc ;
model event=d1-d3 /maxiter=100 rsquare;
strata id;
ods output ParameterEstimates=esti  rsquare=rsq nobs=n;
quit;
data esti;
set esti;
retain t 0;
t=t+1;
if probchisq<0.01 then sig=round(estimate,0.01)||"***";
else if 0.01<probchisq=<0.05 then sig=round(estimate,0.01)||"**";
else if 0.05<probchisq=<0.1 then sig=round(estimate,0.01)||"*";
else sig=round(estimate,0.01)||"";
tv='(' || left ( round(estimate/stderr,0.01)||')');
tv2=(estimate/stderr)**2;
run;
data a1;
set esti;
model=left(sig);
type=1;
keep model t type variable;
run;
data a2;
set esti;
model=tv;
type=2;
keep model t type ;
run;
data rsq;
set rsq;
model=left(round(nvalue1*100,0.01)||"%");
t=100;

```

```
type=1;
variable='R2';
keep model t type variable;
run;
data n;
set n;
model=left(NObsUsed||");
t=1000;
type=1;
variable='N';
if _n_=2 then output;
keep model t type variable;
run;
data final;
set a1 a2 rsq n;
run;
proc sort data=final out=final(keep=model variable);by t type;
run;
```

该部分的程序语法完全依照上一节的介绍，不需要进行额外的修改，conditional Logistic 回归程序执行的结果如图 14-10 所示。

Variable	model
d1	-0.07 (-1.45)
d2	2.86*** (3.7)
d3	2.33** (2.07)
R2	6.73%
N	421

图 14-10

14.3 multinomial logistic regression

本章先前部分所讨论的应变数都属于二元性的变量，但是在真实世界中，管理当局往往会面对三种以上的抉择，例如当公司管理当局决定要发放现金给股东时，可以选择发放现金股利、买回库藏股以及减资，而这三类选择并没有任何优先级，其选项仍然是属于名目尺度，所以可以采用多项式罗吉

斯特回归模型，其方法是在应变量中增列一个对照组 (reference group)，以下以 example_14_3.sas 为例来介绍该语法。多程序事件样本整理的语句如表 14-15 所示。

表 14-15

<pre>data a1; set aa.div_event; /*现金股利事件*/ event=1; y=y-1; drop m d; run; data a2; set aa.repurchase_event; /*买回库藏股或者股票回购事件*/ event=2; y=y-1; drop m d; run; data a3; set aa.event; /*现金减资事件*/ event=3; y=y-1; drop m d; run;</pre>

首先，将数据进行整理，取出现金股利、减资以及库藏股，以作为之后进行 multinomial logistic regression 分析之用。财务数据整理的语句如表 14-16 所示。

表 14-16

<pre>data a; set a1 a2 a3; run; proc sort data=a nodup;by code y; run; libname aa 'D:\The Application of SAS in Financial Research\SAS data\financial'; data b; set aa.financial; if y<1990 then delete;</pre>

续表

```
run;
proc sort data=b;by code y;
run;
data a;
merge a b;by code y;
d1=liquidasset/liquiddebt;
d2=longinvestment/totalasset;
d3=longdebt/totalasset;

if d3>1 or d1=, or d2=, or d3=, or event=, then delete;
keep event d1-d3;
run;
```

此处运用宏程序语法将公司事件进行编码，发放现金股利为 1、买回库藏股为 2、减资为 3，藉此便可以探讨什么原因导致公司使用现金分配，为了研究目的，将没有发生这三种事件的数据皆删除掉。multinomial logistic 回归的语句如表 14-17 所示。

表 14-17

```
proc logistic data=a ;
model event(ref='1')=d1-d3/ maxiter=100 link=glogit;
quit;
```

和先前的程序语法有所不同，此处的语法不再需要 desc 了，反而加了一个声明对照组的语法，不管数据有几组，都会一一跟该组作比较，亦即对照组为 0，与其比较的组别设为 1，而在模型后面还需要加入 link=glogit，否则 SAS 会认定该笔数据是属于顺序性的数据，不会估计出比较性的系数值。multinomial logistic 回归的结果如图 14-11 所示。

Analysis of Maximum Likelihood Estimates						
Parameter	event	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	2	1	-0.1195	0.0253	22.2458	<.0001
Intercept	3	1	-3.0642	0.1823	282.5264	<.0001
d1	2	1	0.000166	0.000583	0.0815	0.7753
d1	3	1	-0.4139	0.0790	27.4414	<.0001
d2	2	1	0.6762	0.0938	51.9567	<.0001
d2	3	1	0.3258	0.3894	0.7001	0.4027
d3	2	1	1.1575	0.1707	46.0061	<.0001
d3	3	1	2.3900	0.6500	13.5217	0.0002

图 14-11

估计的结果如图 14-11 所示，SAS 是依照变量和 event 来排序的，在解释上分别为：相较于事件 1 而言，d2、d3 越高，会增加事件 2 的发生概率；d1 会降低事件 3 的发生概率，d3 越高，会增加事件 3 的发生概率。

行文至此，读者或许会感到困惑，为何不分别将样本区分为含有事件 1 与事件 2 以及含有事件 1 与事件 3 来进行 logistic regression，难道结果会不一样吗？下面我们将删除事件 3 来分析事件 1 与事件 2 的 logistic regression。两事件的 Logistic 回归语句如表 14-18 所示。

表 14-18

data a1;					
set a;					
if event=3 then delete;					
run;					
proc logistic data=a1 desc;					
model event=d1-d3 / maxiter=100;					
quit;					

由于已经确定要以事件 1 作为对照组，因此要计算的是事件 2 的概率，所以仍要加上 desc 要求计算较大数据的概率，两事件 Logistic 回归结果如图 14-12 所示。

Analysis of Maximum Likelihood Estimates					
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-0.1208	0.0254	22.6449	<.0001
d1	1	0.000166	0.000583	0.0815	0.7753
d2	1	0.6796	0.0939	52.4320	<.0001
d3	1	1.1673	0.1710	46.6201	<.0001

图 14-12

结果与使用 multinomial logistic regression 一致，仅以现金增资与买回库藏股而言，当 d2 与 d3 的数值越高，公司越偏好使用库藏股。需要注意的是样本数不同，使用 multinomial logistic regression 是估计三种事件的可能性的比较，分开估计则是估计两种事件的可能性，如果采用分开估计或许能得到相似的结果，但是却不太能够符合实际的状况，因此仍建议使用 multinomial logistic regression 的方法。接下来，在使用宏语法的同时将三种对照结果估计出来。multinomial logistic 宏语句如表 14-19 所示。

表 14-19

%macro multi_logit;					
%do i=1 %to 3;					
proc logistic data=a desc ;					
model event(ref="&i")=d1-d3 /maxiter=100 link=glogit;					
ods output ParameterEstimates=esti ;					

```

quit;
data esti;
set esti;
retain t 0;
if variable^=lag(variable) then t=t+1;
if probchisq<0.01 then sig=round(estimate,0.01)||'***';
else if 0.01<probchisq=<0.05 then sig=round(estimate,0.01)||'**';
else if 0.05<probchisq=<0.1 then sig=round(estimate,0.01)||'*';
else sig=round(estimate,0.01)||'';
tv=(' || left ( round(estimate/stderr,0.01)||')');
tv2=(estimate/stderr)**2;
run;
data a1;
set esti;
model=left(sig);
type=1;
keep response model t type variable;
run;
data a2;
set esti;
model=tv;
type=2;
keep response model t type ;
run;
data final;
set a1 a2 ;
run;
proc sort data=final out=final;by t type variable;
run;
proc transpose data=final out=ref&i (drop=t type _name_);
var model;
id response;
by t type variable;
quit;
%end;
%mend;
%multi_logit;

```

该部分的语法，基本上是通过修改 logistic regression 的程序产生的，修改时模型撰写的语句如表 14-20 所示。

表 14-20

<pre>proc logistic data=a desc ; model event(ref="&i")=d1-d3 /maxiter=100 link=glogit; ods output ParameterEstimates=esti ; quit;</pre>

此部分取消输出观测值以及 R^2 的数据，在此只专心讨论 multinomial logistic regression 的系数值，最主要的原因是，不管是使用何种对照组进行 multinomial logistic regression，其观测值与 R^2 皆相同，因此不输出这两笔数据，读者使用自己的数据进行 multinomial logistic 回归时，可在此修改自己的模型。系数值的排序整理语句如表 14-21 所示。

表 14-21

<pre>data esti; set esti; retain t 0; if variable^=lag(variable) then t=t+1; 以下略</pre>
--

在此处多加了条件句来设定 t，由于进行 multinomial logistic regression，SAS 的输出形式是不同的比较组，再区分不同的系数值，因此截距项会重复 N 个比较组的数据后，再重复 N 比较组的 d1 系数值，其余系数值亦同样进行，为了要让相同的变量有相同的 t，故添加了该语法。

此处将 final 依照 t 排序后，再进行转置，多模型转置处理的语句如表 14-22 所示。先前让相同的变量有同样的 t，就是为了让数据值能够顺利转置成如图 14-13 所示的回归整理结果。

表 14-22

<pre>proc sort data=final out=final;by t type variable; run; proc transpose data=final out=ref&i (drop=t type _name_); var model; id response; by t type variable; quit;</pre>
--

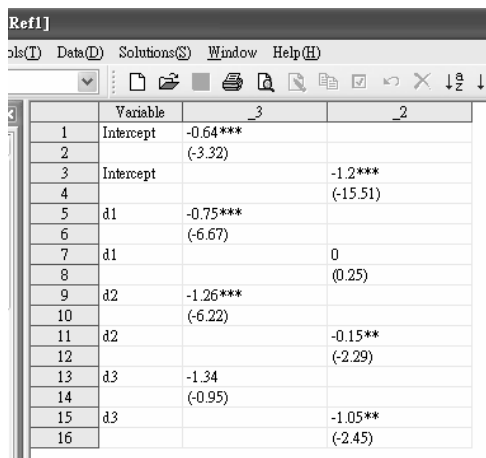
	Variable	_3	_2
1	Intercept	-3.06***	-0.12***
2		(-16.81)	(-4.72)
3	d1	-0.41***	0
4		(-5.24)	(0.29)
5	d2	0.33	0.68***
6		(0.84)	(7.21)
7	d3	2.39***	1.16***
8		(3.68)	(6.78)

	Variable	_3	_1
1	Intercept	-2.94***	0.12***
2		(-16.16)	(4.72)
3	d1	-0.41***	0
4		(-5.24)	(-0.29)
5	d2	-0.35	-0.68***
6		(-0.9)	(-7.21)
7	d3	1.23*	-1.16***
8		(1.91)	(-6.78)

	Variable	_2	_1
1	Intercept	2.94***	3.06***
2		(16.16)	(16.81)
3	d1	0.41***	0.41***
4		(5.24)	(5.24)
5	d2	0.35	-0.33
6		(0.9)	(-0.84)
7	d3	-1.23*	-2.39***
8		(-1.91)	(-3.68)

图 14-13

如果事先不使 t 相同的话，结果则会不相同，未纳入数据 t 变量的整理结果如图 14-14 表示。



	Variable	_3	_2
1	Intercept	-0.64***	
2		(-3.32)	
3	Intercept		-1.2***
4			(-15.51)
5	d1	-0.75***	
6		(-6.67)	
7	d1		0
8			(0.25)
9	d2	-1.26***	
10		(-6.22)	
11	d2		-0.15**
12			(-2.29)
13	d3	-1.34	
14		(-0.95)	
15	d3		-1.05**
16			(-2.45)

图 14-14

虽然仍可理解两种模型的系数值，但是在数据的阅读方面却有极大的不便之处，因此才特别修改该部分语法。在此处要特别注意的是，修改 t 的条件句，是作者在发现上面的问题之后，才回过头去

修改前面程序语法的，在这一过程中仍花费不少时间去寻找其他的可能语法，因此在撰写程序时，都需要反复回过头去检验才有可能完成一个程序的语法。

14.4 分类与概率转换

前面几节探讨了三种不同的财务比率对公司进行现金分配的影响能力，然而，logistic regression 亦有进行分组的做法，此外在破产研究上，由于公司发生破产已经是事后的已知事实，虽然在研究上我们可以放入虚拟变量，但是会犯 looking-ahead bias（事前诸葛亮）的错误，利用 logistic regression 亦可估算出每家公司的模型所预测出来的破产概率，如此一来，可以将事后的虚拟变量转换成事前的连续变量，在本节中以 example_14_4.sas 为例来介绍 proc logistic 中的模型预测估算出来的分类、概率数据，或者是由模型估计出来的参数数值的总和，下面先介绍 logistic regression 的分组。样本数据的整理语句如表 14-23 所示。

表 14-23

```

DM'LOG;CLEAR;OUT;CLEAR';
option nonotes nlabel;
libname aa 'D:\The Application of SAS in Financial Research\SAS data\event date';

data a;
set aa.event;
event=1;
/*要与未发生事件的样本区分*/
y=y-1;
/*因为要以以前的数据预测未来是否会减资，所以在处理上要先减 1*/
run;

proc sort data=a nodupkey;by code y;
run;

libname aa 'D:\The Application of SAS in Financial Research\SAS data\financial';
data b;
set aa.financial;
if y<1990 then delete;
run;

proc sort data=b;by code y;
run;
```

先采用 14.1 节所使用的数据来进行分组的分析，藉由分组，可以计算除了 R^2 之外，另外一种检验 logistic regression 优劣方法的指标。求得 logistic 概率转换的语句如表 14-24 所示。

表 14-24

```
proc logistic data=a noprint;
model event=d1-d3/ maxiter=100;
output out=b predprobs=(i);
quit;
```

要让 SAS 报告出分组后的结果，不需要了解 logistic regression 的回归系数是什么，此时有一个在 proc means 以及 proc reg 重复出现的语法可以使用，output out=filename，为了解模型预测出来的分组状况，在此处要采用 predprobs=(i)的语法，SAS 自然就会报告相关必要的信息。logistic 回归概率转换的结果如图 14-15 所示。

	event	d1	d2	d3	_FROM_	_INTO_	IP_0	IP_1
1	0	0.8065622589	0.8433999748	0.0320059872	0	0	0.9810315506	0.0189684494
2	0	1.5948601866	0.7583573735	0.0553945762	0	0	0.9842264348	0.0157735652
3	0	0.9558454498	0.7231202555	0.0063447034	0	0	0.9788168337	0.0211831663
4	0	1.1253199517	0.81316592	0.0077802622	0	0	0.9813214418	0.0186785582

图 14-15

如图 14-15 所示，SAS 会新增 _from_、_into_ 以及 IP_0、IP_1 这四个变量，前两个变量是固定的，亦即该笔观测值是来自 (_from_) 某组，模型预测值将其放入 (_into_) 某组，而 IP_则表示该数据有多少概率应该是该组别的，由于数据中只有 0 与 1 两组，故产生 IP_0 与 IP_1 两个变量，若有三组，则会根据命名的组别 K，产生 IP_K 等变量。

就本例中，可以如表 14-25 所示的实际数值与预测值的示意表来表示：

表 14-25

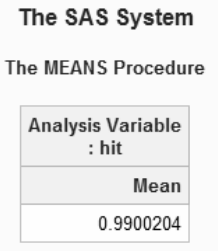
	模型预测 1	模型预测 0
实际组别 1	a	c
实际组别 0	b	d

其中 a、b、c、d 分别为各个组合下的观测值，模型的命中率 (hit ratio) 便为(a+d)/(a+b+c+d)，我们可以藉由撰写如表 14-26 所示的 logistic 回归分组的命中率的语法来加以计算。

表 14-26

```
data b;
set b;
if _from=_into_ then hit=1;else hit=0;
run;
proc means mean;
var hit;
run;
```

由于命中率的分子为 $a+d$ ，这两组的特性为 `_from_ = into_`，我们可以设立虚拟变量 `hit`，令其为 `_from_ = into_` 时为 1，其余为 0，此时若计算 `hit` 的均值时，会等于 $(a+d)/(a+b+c+d)$ 。命中率预测的结果如图 14-16 所示。



The SAS System

The MEANS Procedure

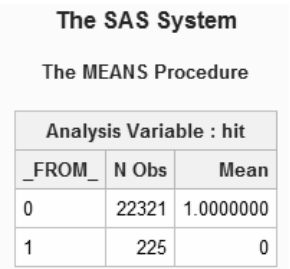
Analysis Variable : hit
Mean
0.9900204

图 14-16

结果显示命中率为 99.00%，似乎是非常良好的模型，实际上仍需要做进一步的检验，因为该笔数据只有少部分的数据会进行减资，因此再进一步分别检验，有进行减资的组别与没有进行减资的组别的正确率。命中率分组计算的语句如表 14-27 所示。样本分配对命中率计算影响的结果如图 14-17 所示。

表 14-27

<code>proc means mean;</code>
<code>var hit;</code>
<code>class _from_;</code>
<code>run;</code>



The SAS System

The MEANS Procedure

Analysis Variable : hit		
FROM	N Obs	Mean
0	22321	1.0000000
1	225	0

图 14-17

由于 0 这组数据过多，使得 logistic regression 完全无法分出 0、1 的差别，这有两种可能原因，第一，样本比率问题，第二，预测变量不佳。将样本缩减为一比一的状况，再次进行 logistic regression，检验其命中率¹。这样的命中率计算的结果如图 14-18 所示。

1 我们不能使用 conditional logistic regression 来进行分组，因为 conditional logistic regression 无法用 `predprobs` 这个语法。

The SAS System		
The MEANS Procedure		
Analysis Variable : hit		
FROM	N Obs	Mean
0	196	0.5918367
1	225	0.6488889

图 14-18

由图 14-18 可见，整体模型的命中率为 64%左右，但是模型仍会预测出 1 的组别，而不像前例中，仅预测 0 的组别，接下来我们介绍如何求出预测的概率值以及模型的预测数据。logistic 回归系数值和与预测概率输出的语法如表 14-28 所示。

表 14-28

proc logistic data=a desc noprint;		
model event=d1-d3/ maxiter=100;		
output out=b xbeta=xbeta pred=p;		
quit;		

在语法中，使用的是 xbeta 以及 pred，xbeta 指的是将每个观测值的 x 值乘上系数值的总和，pred 则是预测出该笔数据为事件的概率是什么。logistic 的系数值和与预测概率如图 14-19 所示。

	event	d1	d2	d3	id	_LEVEL_	xbeta	p
1	1	0.9492720691	0.1571815234	0.2971854905	1	1	0.8244695765	0.6951842826
2	1	0.1854215927	0.0987106003	0	2	1	-0.030538592	0.4923659453
3	1	1.0898012642	0.0783965881	0.0240088322	3	1	-0.10323659	0.4742137505
4	1	0.3250263005	0.3775130127	0.3307440525	4	1	1.5936300934	0.831126219
5	1	0.3058815698	0.4067754221	0.4138595469	5	1	1.8891259849	0.8686558438
6	1	0.7208272031	0.1504244671	0.1602045566	6	1	0.4781039862	0.6173000589
7	1	0.1038050231	0.1250242525	0.0607738193	7	1	0.205253147	0.551133895
8	1	0.8845685435	0.1317352109	0.0015204416	8	1	0.0101487741	0.5025371717
9	1	0.6542304488	0.1240557021	0	9	1	0.0033597598	0.5008399392
10	1	0.5820393582	0.0045605581	0	10	1	-0.333767371	0.4173242471
11	1	0.6919812085	0.3844275597	0	11	1	0.7478507706	0.6787102126
12	1	0.314331583	0.2688339849	0.2063849736	12	1	0.9684255113	0.7248055576
13	1	0.4931063512	0.1695364682	0.1893311895	13	1	0.6254167949	0.6514495093
14	1	0.4447074131	0.0653956964	0.0820092651	14	1	0.0593925649	0.514843778
15	1	0.789296812	0.2417388013	0	15	1	0.3300698432	0.5817763707
16	1	0.7211638758	0.2892258479	0.0038858634	16	1	0.4818849792	0.6181928863

图 14-19

当用户下了 xbeta 或者 pred 的语法之后，SAS 会自动产生 _LEVEL_ 的变量，目的在于提醒用户 xbeta 以及 pred 是计算 0 还是 1 的概率，若 _LEVEL_ 为 1，则表示该笔观测值的 xbeta 转换为概率是什么，就第一笔数据而言，xbeta 为 0.8244，其会发生减资的概率为 69.51%，根据前一节的概率转换，概率的转换为 $\pi = e^{\alpha + \beta x} / (1 + e^{\alpha + \beta x})$ ，因此可以进一步进行验算。系数值总和转换验算的语法如表 14-29 所示。系数值总和转换验证的结果如图 14-20 所示。

表 14-29

```
data b;
set b;
prob=exp(xbeta)/(1+exp(xbeta));
run;
```

event	d1	d2	d3	id	_LEVEL_	xbeta	p	prob
1	0.9492720691	0.1571815234	0.2971854905	1	1	0.8244695765	0.6951842826	0.6951842826
1	0.1854215927	0.0987106003	0	2	1	-0.030538592	0.4923659453	0.4923659453
1	1.0896012642	0.0783965881	0.0240088322	3	1	-0.10323659	0.4742137505	0.4742137505
1	0.3250263005	0.3775130127	0.3307440525	4	1	1.5936300934	0.831126219	0.831126219
1	0.3058815698	0.4067754221	0.4138595469	5	1	1.8891259849	0.8686559438	0.8686559438
1	0.7208272031	0.1504244671	0.1602045566	6	1	0.4781039862	0.6173000589	0.6173000589
1	0.1038050231	0.1250242525	0.0607738193	7	1	0.205253147	0.551133895	0.551133895
1	0.8845685435	0.1317352109	0.0015204416	8	1	0.0101487741	0.5025371717	0.5025371717

图 14-20

经验算，两者概率是一模一样的，显示 pred 确实是计算该概率的选项，这样就取得了概率数据与模型计算出来的 xbeta，在财务危机研究中，在报酬率或者持股的模型中加入该变量，亦可视为风险溢价或者考虑了破产风险的投资决策，由于该变量采用滞后后期的数据来处理，因此一般不会有 looking ahead bias 的问题。

14.5 总结

本章介绍了三种 logistic regression 的 SAS 程序语法。在财务研究中，logistic regression 是一个重要的实证方法，在探讨基金经理人升迁或者公司财务危机时，是必备的研究工具，本章利用减资、库藏股以及现金股利的议题来介绍 SAS 程序，在学术研究上较少见，仅仅是为了介绍程序便利而使用的，并非是正统的研究方法。读者可根据其研究目的，适当地修改本章所附的程序语法，便能对你的实证有相当的帮助。

第 15 章

tobit 模型（proc lifereg）

一般的最小平方估计是为连续性且数据范围没有特定限制的数据，而 logit 以及 probit model 被解释变量为类别变量，采用的是特殊的概率估计方式。然而本章所要介绍的是被解释变量的数据类型属于受限制以及连续的，根据其形式，又可以分为受限（censored）以及截断（truncated）两种，在 SAS 里则提供了 proc lifereg 来加以分析¹。

¹ 本质上，lifereg 是最适合用来分析存活数据的，在进行存活分析时，亦可使用 proc lifereg 来分析，详细的语法可自行参阅 SAS 的 HELP。

15.1 受限数据 (censored data) 与截断数据 (truncated data)

假设有一连续性随机变量 Y 以及常数 C ，但是由于某些特殊原因，当数据 Y 大于或小于 C 时，则数据便无法收集与登录，例如鸿海公司营业收入首次破兆元台币时曾发生系统里数据的单元格式不够用，无法正确记载数据的问题¹，所幸后来问题得到解决，因此仍可正确记载数据，若一旦无法登录数据，则该笔兆元台币营业收入的数据将会是以缺失值 (missing value) 来存储，或者皆以 999,999,999,999 来登录超过兆元的数据，那么这样的数据称为截断数据，又称为右受限数据，因为该数据是在右边被截断；而在学期成绩中，也有可能因为老师不想害人，而将低于 60 分的成绩全部往上调到 60 分的，但是此处的数据并非是左受限数据，由于分数不可能高于 100 分，所以学期成绩是一个双边受限制的数据²；而左受限的数据最常见的就是取对数的数据，因为对数值必为大于 0 的值，如果其原始数据小于 0 就无法计算。

在 SAS 中，针对 tobit 的模型有两种估计方法，第一种是针对右受限的数据进行估计，这主要是整体而言右尾截断的数据较为常见 (并非针对财务数据而言)，第二种则是较为一般化的语法，但无论哪一种方法，在使用 tobit 模型时，在被解释变量的部分，必须使用两个变量才能进行估计，其中一个数据是原始数据的变量，另外一个辅助用的变量。

以下以 example_15_1.sas 为例来介绍该语法³。生成断尾数据的语句如表 15-1 所示。

表 15-1

```
data subset;
  input Hours Yrs_Ed Yrs_Exp @@;
  if Hours eq 0
    then Lower=.;
    else Lower=Hours;
datalines;
0 8 9 0 8 12 0 9 10 0 10 15 0 11 4 0 11 6
1000 12 1 1960 12 29 0 13 3 2100 13 36
3686 14 11 1920 14 38 0 15 14 1728 16 3
1568 16 19 1316 17 7 0 17 15
;
run;
```

- 1 其原因是由于计算机公司从未设想过台湾地区会有公司的营业收入破兆元台币，因此没有设立兆元台币的单元格式。
- 2 实际上，原始的学期成绩就是双边受限制的数据，因为最低分为 0 分，最高分为 100 分。
- 3 由于缺乏适当的实例，以下的数据截取于 SAS 内建的 Example 48.2。

该数据为 Mroz (1987) 的部分研究数据，主要是探讨妇女工作时数与其教育程度和过去工作经历的关系，数据读取后会呈现如图 15-1 所示的断尾数据结果。

	Hours	Yrs_Ed	Yrs_Exp	Lower
1	0	8	9	.
2	0	8	12	.
3	0	9	10	.
4	0	10	15	.
5	0	11	4	.
6	0	11	6	.
7	1000	12	1	1000
8	1960	12	29	1960
9	0	13	3	.
10	2100	13	36	2100
11	3686	14	11	3686
12	1920	14	38	1920
13	0	15	14	.
14	1728	16	3	1728
15	1568	16	19	1568
16	1316	17	7	1316
17	0	17	15	.

图 15-1

由于有部分的妇女工作时数皆为 0，因此该数据是一种左尾受限的数据，另外产生一些变量 lower，当该数据为 0 时，令其为缺失值，其余则等于原来的数据。接下来介绍 tobit 模型的语法如表 15-2 所示。

表 15-2

<pre>proc lifereg data=subset ; model (lower, hours) = yrs_ed yrs_exp / d=normal;quit;</pre>
--

由模型所示，原本的被解释变量变为 (lower, hours) 这两个变量组合而成，其中 hours 的变量是完整性的数据且其被放置在右边，lower 具有缺失值，且被放置在左边，显示该被解释变量的数据是一个左受限的数据。亦即 0 小时之下的数据皆不会存在，可将 0 视为缺失值的门槛，接下来便说明如图 15-2 所示的 tobit 回归模型的数据。

The LIFEREG Procedure	
Model Information	
Data Set	WORK.SUBSET
Dependent Variable	Lower
Dependent Variable	Hours
Number of Observations	17
Noncensored Values	8
Right Censored Values	0
Left Censored Values	9
Interval Censored Values	0
Name of Distribution	Normal
Log Likelihood	-74.9369977

图 15-2

首先 SAS 会先确认数据特性，在 17 笔数据中，有几笔是右受限的数据，有几笔是左受限的数据，有几笔是不受限的数据，接着确定该笔数据的分配形式，该分配是由于在模型中撰写 d (distribution 之意) =normal 而来的。接下来估计结果。tobit 回归模型的系数值如图 15-2 所示。

Parameter	DF	Estimate	Standard Error	95% Confidence Limits	Chi-Square	Pr > ChiSq
Intercept	1	-5598.64	2850.248	-11185.0 -12.2553	3.86	0.0495
Yrs_Ed	1	373.1477	191.8872	-2.9442 749.2397	3.78	0.0518
Yrs_Exp	1	63.3371	38.3632	-11.8533 138.5276	2.73	0.0987
Scale	1	1582.870	442.6732	914.9433 2738.397		

图 15-2

在系数值估计中，多出了 scale 这个参数，该变量为估计所得的预测值的标准偏差，其功能主要是用来求得在 tobit model 中受限制时的预测值，而 tobit model 通常也提供两阶段回归模型的预测变量的估计。接下来，介绍如何利用 Tobit model 求得预测值。求算 tobit 回归模型系数值的语句如表 15-3 所示。

表 15-3

proc lifereg data=subset noprint outest=outest(keep=_scale_);
model (lower, hours) = yrs_ed yrs_exp / d=normal;
output out=a(drop=censor _prob_) xbeta=xbeta;
quit;
data a;
set a;
if _n_=1 then set outest;
lambda = pdf('NORMAL',Xbeta/_scale_)/ cdf('NORMAL',Xbeta/_scale_);
Predict = cdf('NORMAL', Xbeta/_scale_)* (Xbeta + _scale_*lambda);
label Xbeta='MEAN OF UNCENSORED VARIABLE'
Predict = 'MEAN OF CENSORED VARIABLE';
drop lambda _scale_;
run;

在该语法中 Xbeta 输出的数值就是 $X'\beta$ 的数值，然而该模型的数值是未受限变量的预测值，要计算受限变量的预测值，则需要使用 CDF 以及 PDF 的数值加以计算，如图 15-3 所示为求得 tobit 回归模型预测值后的结果。

从结果来看，当考虑了左受限的情况后，预测变量的数值都会调整到大于 0 的状况，该结果比较符合实际情况的预测，在取得该数值之后，可以将该变量纳入其他的模型进行第二阶段的回归。我们在简短介绍了左受限的模型之后，接着探讨双边受限的模型。产生报酬率模型数值的语句如表 15-4 所示。

	Hours	Yrs_Ed	Yrs_Exp	Lower	MEAN OF UNCENSORED VARIABLE	MEAN OF CENSORED VARIABLE
1	0	8	9	.	-2043.422603	73.461074002
2	0	8	12	.	-1853.41126	94.231403981
3	0	9	10	.	-1606.937778	128.1031892
4	0	10	15	.	-917.1044961	276.04498808
5	0	11	4	.	-1240.66504	195.76292919
6	0	11	6	.	-1113.990812	224.71570134
7	1000	12	1	1000	-1057.528672	238.62717089
8	1960	12	29	1960	715.91052316	1052.9382799
9	0	13	3	.	-557.7067326	391.41641394
10	2100	13	36	2100	1532.4180328	1672.4967456
11	3686	14	11	3686	322.13789113	805.57516986
12	1920	14	38	1920	2032.2399719	2106.8078251
13	0	15	14	.	885.29694436	1170.3932151
14	1728	16	3	1728	561.73640014	951.69482009
15	1568	16	19	1568	1575.1302258	1708.2403869
16	1316	17	7	1316	1188.2325675	1395.6081593
17	0	17	15	.	1694.9294803	1809.9654397

图 15-3

表 15-4

```
data a;
do i=1 to 15;
x1=rannor(1);
x2=rannor(2);
y=2+3*x1+2*x2+rannor(3);
output;
end;
run;
```

在本例中，以 1 以及 7 作为左右截断的门槛，亦即产生一个左右受限的数据模式，但是要注意的是，由于受限数据表示观察不到 7 以上的数据，且也观察不到 1 以下的数据，此处需要做额外的处理。产生双边受限的数值如表 15-5 所示。

表 15-5

```
/*假设左边限制为 1，右边限制为 7*/
data a;
set a;
if y<=1 then y=1;
if y>=7 then y=7;
if y=1 then lower=.;
else lower=y;
/*产生左受限数据*/
if y=7 then higher=.;
else higher=y;
/*产生右受限数据*/
run;
```

由此可以产生两个数据, 其中 lower 是针对左受限模型产生数据, higher 是针对右受限模型产生数据, 藉此可以使用三种模型, 其中 (lower, y) 为左受限模型, (y, higher) 为右受限模型, 而使用 (lower, higher) 则是双边受限模型, 双边受限模型产生数据的结果如图 15-4 所示。

	i	x1	x2	y	lower	higher
1	1	1.8048229506	-0.079915021	7	7	.
2	2	-1.083317655	2.2382943651	2.602403471	2.602403471	2.602403471
3	3	0.5136577083	-0.086609117	2.7735761577	2.7735761577	2.7735761577
4	4	0.0318908181	-0.737798572	1	.	1
5	5	0.6850047647	-0.804158132	1.7024169507	1.7024169507	1.7024169507
6	6	-0.795502822	0.3407105493	1	.	1
7	7	-1.349846516	0.432704861	1	.	1
8	8	1.4251270104	-0.415801019	7	7	.
9	9	-1.057726424	-0.948332672	1	.	1
10	10	0.3919789416	-0.076141279	4.2442111954	4.2442111954	4.2442111954
11	11	-0.630841419	-0.635758247	1	.	1
12	12	-0.076283637	0.9653583786	2.4851674542	2.4851674542	2.4851674542
13	13	1.1844900022	-0.343693949	5.9563248954	5.9563248954	5.9563248954
14	14	-0.135313068	-1.359501933	1	.	1
15	15	-0.40968631	0.6542014011	2.4786009692	2.4786009692	2.4786009692

图 15-4

从数据来看, lower 的数据皆为低于某个数值后便为缺失值, higher 为高于某个数值后便为缺失值, 而且读者会注意到数据中会大量出现数据为 1 的数值, 以及 2 个数据为 7 的数据而与真实的数据不同, 这是由于受限制的数据产生的状况, 1 以下的数据完全无法登录, 7 以上的数据也只能登录为 7。接下来, 在采取正确的模型之前, 先采用 (y, lower) 来进行模型估计, 将数值错置的示范语法如表 15-6 所示。

表 15-6

```

Proc lifereg data=a 297utset=297utset(keep=_scale_);
model (y,lower)=x1 x2/d=normal;
output out=a(drop=censor _prob_) xbeta=xbeta;
quit;
data a;
set a;
if _n_=1 then set outest;
lambda = pdf('NORMAL',Xbeta/_scale_)/cdf('NORMAL',Xbeta/_scale_);
Predict = cdf('NORMAL', Xbeta/_scale_)*(Xbeta + _scale_*lambda);
label Xbeta='MEAN OF UNCENSORED VARIABLE'
       Predict = 'MEAN OF CENSORED VARIABLE';
drop lambda _scale_;
run;

```

数值错置后的系数估计出来的预测值结果如图 15-5 所示。

	i	x1	x2	y	lower	higher	MEAN OF UNCENSORED VARIABLE	MEAN OF CENSORED VARIABLE
1	1	1.8048229506	-0.079915021	7	7	.	6.2148955419	6.2148955481
2	2	-1.083317655	2.2382943651	2.602403471	2.602403471	2.602403471	2.3023361114	2.3119684871
3	3	0.5136577083	-0.086609117	2.7735761577	2.7735761577	2.7735761577	4.204642529	4.2046775337
4	4	0.0318908181	-0.737798572	1	1	.	3.292176976	3.2928750225
5	5	0.6850047647	-0.804158132	1.7024169507	1.7024169507	1.7024169507	4.2915696263	4.2915951807
6	6	-0.795502822	0.3407105493	1	1	.	2.2750431907	2.2853073683
7	7	-1.349846516	0.432704861	1	1	.	1.4357178622	1.4938123065
8	8	1.4251270104	-0.415801019	7	7	.	5.540145443	5.5401455987
9	9	-1.057726424	-0.948332672	1	1	.	1.5444306125	1.5919436006
10	10	0.3919789416	-0.076141279	4.2442111954	4.2442111954	4.2442111954	4.0179756514	4.0180432558
11	11	-0.630841419	-0.635758247	1	1	.	2.2867538627	2.2967425938
12	12	-0.076283637	0.9653583786	2.4851674542	2.4851674542	2.4851674542	3.5502527737	3.550569722
13	13	1.1844900022	-0.343693949	5.9563248954	5.9563248954	5.9563248954	5.1838536806	5.183854427
14	14	-0.135313068	-1.359501933	1	1	.	2.8764368518	2.8787070962
15	15	-0.40968631	0.6542014011	2.4786009692	2.4786009692	2.4786009692	2.953708412	2.9555474392

图 15-5

接下来采用 (lower, y) 进行估计, 采用正确顺序的回归预测值语法如表 15-7 所示, 还可以比较两次估计的结果。

表 15-7

data a;	
set a;	
keep i y x1 x2 higher lower;	
run ;	
proc lifereg data=a outest=outest(keep=_scale_);	
model (lower,y)=x1 x2/d=normal;	
output out=a(drop=censor _prob_) xbeta=xbeta;	
quit ;	
data a;	
set a;	
if _n_=1 then set outest;	
lambda = pdf('NORMAL',Xbeta/_scale_)/ cdf('NORMAL',Xbeta/_scale_);	
Predict = cdf('NORMAL', Xbeta/_scale_)* (Xbeta + _scale_*lambda);	
label Xbeta='MEAN OF UNCENSORED VARIABLE'	
Predict = 'MEAN OF CENSORED VARIABLE';	
drop lambda _scale_;	
run ;	

采用正确顺序的回归预测值结果如图 15-6 所示。

	i	x1	x2	y	lower	higher	MEAN OF UNCENSORED VARIABLE	MEAN OF CENSORED VARIABLE
1	1	1.8048229506	-0.079915021	7	7		7.9366687941	7.9366687941
2	2	-1.083317655	2.2382943651	2.602403471	2.602403471	2.602403471	2.5271951627	2.5275386751
3	3	0.5136577083	-0.086609117	2.7735761577	2.7735761577	2.7735761577	3.4120088419	3.4120141951
4	4	0.0318908181	-0.737798572	1			0.4138091184	0.5844044973
5	5	0.6850047647	-0.804158132	1.7024169507	1.7024169507	1.7024169507	2.5616906896	2.5619881438
6	6	-0.795502822	0.3407105493	1			-0.299100093	0.2092613394
7	7	-1.349846516	0.432704861	1			-2.050127433	0.0021692194
8	8	1.4251270104	-0.415801019	7	7		5.9317967682	5.9317967682
9	9	-1.057726424	-0.948332672	1			-3.818295119	5.6761467E-7
10	10	0.3919789416	-0.076141279	4.2442111954	4.2442111954	4.2442111954	3.0080192373	3.0080596737
11	11	-0.630841419	-0.635758247	1			-1.695625543	0.0071626507
12	12	-0.076283637	0.9653583786	2.4851674542	2.4851674542	2.4851674542	3.4751264493	3.4751302798
13	13	1.1844900022	-0.343693949	5.9563248954	5.9563248954	5.9563248954	5.2366554226	5.2366554226
14	14	-0.135313068	-1.359501933	1			-1.425810913	0.0161430185
15	15	-0.40968631	0.6542014011	2.4786009692	2.4786009692	2.4786009692	1.6819327649	1.689411789

图 15-6

从两次结果来看，采用 (lower, y) 的受限预测值的误差较小，显示若模型撰写错误，将是估计上的问题，为此本书建议的撰写方法为，产生左受限变量命名为 lower，右受限变量命名为 higher，或者分别命名为 left 以及 right，如此在撰写模型时，一律将 lower 或者 left 放在括号左边的位置，higher 以及 right 放在括号右边的位置，便可以避免错误。Tobit 回归模型的标准因变量写法如表 15-8 所示。tobit 模型的数据受限结果如图 15-7 所示。

表 15-8

proc lifereg data=a ;	
model (lower,higher)=x1 x2/d=normal;	
quit;	
The LIFEREG Procedure	
Model Information	
Data Set	WORK.A
Dependent Variable	lower
Dependent Variable	higher
Number of Observations	15
Noncensored Values	7
Right Censored Values	2
Left Censored Values	6
Interval Censored Values	0
Name of Distribution	Normal
Log Likelihood	-9.893545752

图 15-7

在这种情况下，SAS 会自动辨认出数据具有左右双重受限的情况，并自动采取适合的模型；同时研究者也需注意：在 Tobit 模型中也有区间受限 (Interval Censored data) 的数据，也就是说被解释变量并非是一个固定不变的数据，而是介于某个区间的数据，在财务研究中这样的数据是出现在什么情况下的呢？将分别介绍如下。

本质上，财务研究中确实被解释变量为一个区间的估计数据，我们以分析师的财务预测为例，来解释其数据呈现在左受限、右受限以及区间受限的型态。当我们进行编码分析师的预测时，常会出现以下几种情况，第一，分析师预测该公司的 EPS 为 0.95，则该笔数值的意义表示 (lower, higher)

= (0.95, 0.95), 其为非受限的数据; 第二, 分析师预测该公司的 EPS 最少为 0.95, 由于其分析最少为多少, 表示其为左受限数据, 因此 (lower, higher) = (. , 0.95); 第三, 分析师预测该公司的 EPS 最多为 0.95, 则表示其为右受限的数据, (lower, higher) = (0.95, .); 最后, 分析师预测该公司的 EPS 介于 0.92 到 0.95 之间, 其 (lower, higher) = (0.9, 0.95)。最后的例子, 即是一种区间受限的表达方式。SAS 在判别受限数据的情况时, 采用双边受限的数据进行解释, 如表 15-9 所示。

表 15-9

Lower	higher	Comparison	Interpretation	situation
非缺失值	非缺失值	相等	非受限数据	5
非缺失值	非缺失值	lower < higher	区间受限数据	介于 5~6 间
缺失值	非缺失值		higher 视为左受限的值	高于 5
非缺失值	缺失值		lower 视为右受限的值	低于 5
非缺失值	非缺失值	lower > higher	该笔数据不可使用	可能是编码错误
缺失值	缺失值		该笔数据不可使用	

由上表可见, 读者在撰写 tobit 模型时候需注意, 避免因些微的疏失而导致估计出来的结果不正确的情况, 在 15.2 节中, 将介绍笔者所撰写的标准化的宏, 提供一种进行 tobit 模型估计的便利方法。

15.2 格式化 tobit 模型输出

本节提供了 tobit 模型的宏语法, 由于 tobit 模型的特殊性, 以下以 example_15_2.sas 为例进行介绍。用 tobit 回归语法进行数据整理的语句如表 15-10 所示。

表 15-10

<pre>data a; input Hours Yrs_Ed Yrs_Exp @@; if Hours eq 0 then Lower=.; else Lower=Hours; censor=hours^=0; datalines; 0 8 9 0 8 12 0 9 10 0 10 15 0 11 4 0 11 6 1000 12 1 1960 12 29 0 13 3 2100 13 36 3686 14 11 1920 14 38 0 15 14 1728 16 3 1568 16 19 1316 17 7 0 17 15 ; run;</pre>
--

在此处，仍然采用 15.1 节的范例，接下来再执行本节撰写的宏，这个宏语法有两个目的，第一，输出受限制模型下的预测值，以提供进行二阶段回归分析的变量；第二，输出 tobit 模型的参数估计值。tobit 回归语句使用的宏语句如表 15-11 所示。

表 15-11

```
%include 'D:\The Application of SAS in Financial Research\CH15\program_cn\tobit.sas';
%tobit(lower,higher,Yrs_Ed Yrs_Exp,3);

proc export data=tobit
outfile="D:\The Application of SAS in Financial Research\CH15\EXCELoutput\tobit"
dbms=xlsx
replace;
sheet='model1';
quit;
```

使用方法如同前面章节的介绍，先呼叫该宏程序，接下来撰写%tobit (lower, higher, x, bit) 等规律，其中 lower 填入左受限或者完整变量，higher 填入右受限或者完整变量，x 指的是填入解释变量，bit 则是指定小数点位数。宏执行完毕时，原有的文件 a 会产生三个变量，即 xbeta、predict 以及 lambda，其中 lambda 即为 inverse Mills ratio，可用于进行 Heckman two stage model 样本选择的模型。

参数估计结果则会输出到 tobit 这个文档，读者在执行程序完毕以后，可以直接将该结果输出，并选择自己喜欢的文档名和工作表名称。Tobit 宏语法的使用如表 15-12 所示。

表 15-12

```
data a;
do i=1 to 15;
x1=rannor(1);
x2=rannor(2);
y=2+3*x1+2*x2+rannor(3);
output;
end;

run;

data a;
set a;
if y<=1 then y=1;
if y>=7 then y=7;
if y=1 then lower=.;
else lower=y;
if y=7 then higher=.;
else higher=y;

run;
```

```
%tobit(lower,y,x1 x2,3);  
proc export data=tobit  
outfile="D:\The Application of SAS in Financial Research\CH15\EXCELOutput\tobit"  
dbms=xlsx  
replace;  
sheet='model2';  
quit;  
%tobit(y,higher,x1 x2,3);  
proc export data=tobit  
outfile="D:\The Application of SAS in Financial Research\CH15\EXCELOutput\tobit"  
dbms=xlsx  
replace;  
sheet='model3';  
quit;  
%tobit(lower,higher,x1 x2,3);  
proc export data=tobit  
outfile="D:\The Application of SAS in Financial Research\CH15\EXCELOutput\tobit"  
dbms=xlsx  
replace;  
sheet='model4';  
quit;
```

在此处连续执行了三次宏语法，进行左受限、右受限以及双边受限模型，a 文件会保留最后一次执行程序的 xbeta、predict 以及 lambda，读者若有需要，在执行下一次宏前，必须要将变量加以修改或者生成别的文档以供使用。

15.3 总结

本章节所介绍的 tobit 模型的篇幅虽然不多，但其在应用上却相当普遍，其特性是能将缺失值转换成连续变量，其最困难之处，并不在于估计参数，而是如何决定和编码（lower, higher）这个特殊的被解释变量，读者在应用上，应特别注意这两个变量的定义，并免估计错误。

第 16 章

事件研究法

本章介绍撰写事件研究法（Event Study）程序的方法，事件研究法为财务会计领域必备的研究实证方法，主要是根据异常报酬率的衡量来评判市场是否对特定事件产生反应，并藉以检验效率市场的假说，而在异常报酬率的衡量上，短期以市场模型、指数模型以及平均调整模型来检验，长期异常绩效的检验则可使用日历期间投资组合法以及买进持有异常报酬率法来检验，本章分别依照这 3 个议题介绍相关程序的撰写。

16.1 短期事件研究法

实证中的事件研究法可了解股价与特定事件间之关联性，主要是探讨特定事件发生时，股价是否有异常变动，进而产生异常报酬率（abnormal return; AR），它适用于检验市场是否为半强式效率市场以及研究其信息内涵（information content）；事实上，亦有学者用交易量来检验信息内涵，若交易量产生异常变动，则会产生异常交易量（abnormal volume）或者异常周转率（abnormal turnover; ATV）。Chen, Cheng, and Gao（2005）便以异常交易量来检验盈余信息的内涵，Chae（2005）的研究则是检验异常交易量、信息不对称以及信息择时之间的关系。Bailey, Li, Mao, and Zhong（2003）同时研究颁布公平揭露计算异常报酬率与异常交易量检验（regulation fair disclosure）之后对各种盈余新闻影响的程度。由于异常交易量的估计方式与异常报酬率相似，所以以下的介绍以异常报酬率为主。

有关事件研究法的步骤，主要有三，第一，先确立事件日；第二，定义和估计异常报酬率；第三，分析结果。

估计期与事件期

事件窗口示意图如图 16-1 所示。由图可见事件窗口横坐标主要分为估计期和事件期两个部分

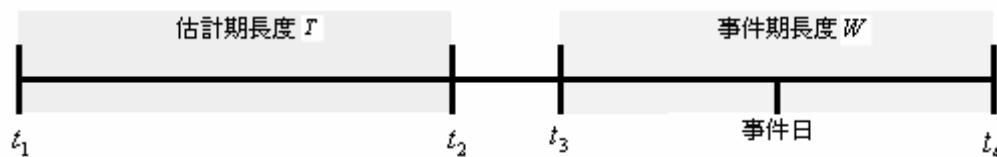


图 16-1

由于要弄清楚证券预期报酬率是什么，因此必须根据一段时间至 t_2 来建立预期模式，此一区间称为估计期，且估计期长度共计 T 期， $T = t_2 - t_1 + 1$ 。以此估计建立其股票报酬率的预期模式，预测可能会受到事件影响的事件期间（ t_3 至 t_4 ），亦即事件期共计 W 期， $W = t_4 - t_3 + 1$ 。在事件期中，以实际报酬率减去预期报酬率，即可得到每一事件期受事件影响所产生的异常报酬率。一般来说，常将事件日定义为第 0 期，事件日前 1 期定义为 -1 期，前 2 期定义为 -2 期；事件日后期则为 +1 期，后 2 期为 +2 期等，依次类推。

定义与估计异常报酬率：在估计异常报酬率之前，要先估计期望报酬率（ $E[R_{it} | X_i]$ ）的方法，常见的有三种方法，方法一为固定均值模型， $E[R_{it} | X_i] = \mu_i$ ，就是假设个股在窗口期间的报酬率等于估计期间的平均报酬率；方法二为市场指数模型， $E[R_{it} | X_i] = R_{mt}$ ，假设正常报酬率是市场当天的报酬率，该方法相对而言比较简便，完全不需要考虑估计期；方法三为市场模型， $E[R_{it} | X_i] = \alpha_i + \beta_i R_{mt}$ ，假设预期报酬率与市场报酬率有一定的关系，市场模式是以估计期的数据，利用普通最小平方法（Ordinary Least Square；简称 OLS）建立以下的回归模式

$$R_{it} = \alpha_i + \beta_i R_{mt} + \varepsilon_{it},$$

其中， ε_{it} 为误差项且 $\varepsilon_{it} \sim N(0, \sigma^2)$ 。经过最小平方法即可以得到估计值 $\hat{\alpha}_i$ 与 $\hat{\beta}_i$ ，因此事件期 E 的

预期报酬率为

$$E(\hat{R}_{iE}) = \hat{\alpha}_i + \hat{\beta}_i R_{mE}$$

检定方法：检定事件是否影响股价或交易量的检定时，文献上大致有参数估计、非参数估计以及拔靴法三种方法，相关的检定可利用 SAS 本身内建的 proc univariate 来检定，拔靴法则是 case by case 的方法，本书难以详尽介绍，有关程序撰写的部分会在本章的最后一节介绍，下面以 example_16_1.sas 为例来介绍短期的事件研究法语法。股票报酬率的整理语句如表 16-1 所示。

表 16-1

libname aa 'D:\The Application of SAS in Financial Research\CH16\data';
data ret;
set aa.daily_return;
run ;
proc sort data=ret;by year month day;
run ;

首先读取股票报酬率数据，并将股票报酬率依照年月日的顺序排好，实际上也可以采用日期进行排序，就依照数据的形态进行改写即可。大盘报酬率整理的语句如表 16-2 所示。

表 16-2

data a;
set aa.drmrf;
run ;
/*
若要使用 3 因子、4 因子、甚至 5 因子模型来进行异常报酬率的检定，
则在此处可以与 SMB HML MOM 等报酬率合并
*/
proc sort data=a nodupkey; by year month day;
run ;
data a;
set a;
t=_n_;
run ;
data t;
set a;
if t=lag(t) then delete;
keep year month day t;
run ;

这个步骤是整理大盘的报酬率，且由于大盘的数据可以帮助我们计算所有交易日的顺序，所以此

处也重新计算一个文档 t 以及一个新变量 t，给予每个交易日有一个连续的数值，该数据可以提供之后的程序整理之用，是一个相当重要的方法，请一定要切记。个股与大盘报酬率合并的语句如表 16-3 所示。

表 16-3

<pre>data ret; merge ret a;by year month day; if t=. then delete; keep year month day ret stkcd rm t; run;</pre>
--

此处将股票报酬率数据与大盘数据合并，以便于之后做三种异常报酬率的估计所需要的数据，接下来便是读取事件日的数据了，在读取事件研究法的研究中，常会读到如果事件日当天股票没有交易，则以接下来的第一个交易日作为事件日，因此在程序上需要对此进行处理。读入事件样本的语句如表 16-4 所示。

表 16-4

<pre>data event; length stknm \$30. stkcd \$6. com_date 8. ann_date 8.; infile 'D:\The Application of SAS in Financial Research\CH16\data\event.txt' firstobs=2 dlm='09'x missover; input type \$ dec \$ cur sn stknm \$ com_date ann_date; year=int(ann_date/10000); month=mod(int(ann_date/100),100); day=mod(ann_date,100); stkcd=substr(stknm,1,6); if ann_date^=.; keep stkcd com_date ann_date year month day; /* 保留 stkcd 股票代码 com_date 完成日 ann_date 事件声明日 year month day */ run; proc sort data=event;by year month day; run; data event;</pre>
--

续表

```

Merge event t;by year month day;

run;

/*
接下来要修正，如果事件宣布日是非交易日
就要将时间修改为下一个交易日

*/

proc sort data=event;by descending year descending month descending day;
run;
data event;
set event;
retain nt 0;
if t^=., then nt=t;
t=nt;
if stked=" then delete;
drop nt;
run;
data event ;
set event;
event=_n_;
run;
```

接着将所需的事件样本读进来，并且设定有关估计期、估计期与事件窗口的间隔时间以及窗口期间，在程序中，初始设定是估计期（est）250 天，间隔（drop）0 天，窗口开前（before）10 天以及开后（after）10 天。取得估计期报酬率数据的语句如表 16-5 所示。

表 16-5

```

%let est=250;
%let drop=0;
%let before=10;
%let after=10;
/*
此处可以自行修改
est 是估计期长短
drop 是估计期与事件期要间隔几天
before 是事件窗口往前开几天
after 是事件窗口往后开几天
*/
```

续表

```
PROC SQL;  
CREATE table a  
as select  
a.stkcd,  
a.event,  
b.ret,  
b.rm  
/*  
若要用三因子、四因子、甚至五因子，  
可在这边抓取相关变量  
*/  
from event a, ret b  
where (a.stkcd=b.stkcd and  
a.t-&est-&drop-&before <=b.t<= a.t-&drop-&before-1)  
order by a.event,a.stkcd;  
quit;
```

以上程序执行完毕可以求得估计期的数据，在计算市场模型以及平均调整模型时都需要用这里的数据，因此在取得该部分的数据后，就可以直接计算。计算估计期市场模型系数值的语句如表 16-6 所示。

表 16-6

```
proc reg data=a adjrsq outest=b noprint;  
model ret=rm;  
/*  
可在这边建构 3、4、5 因子模型  
*/  
by event stkcd;  
quit;  
proc means noprint data=a;  
var ret;  
by event stkcd;  
output out=c(drop=_type_ _freq_) mean=mean;  
run;
```

第一步先进行市场模型的回归式，利用该回归式，就可以进入事件窗口，计算市场模型的异常报酬率。同时在此处，计算各个事件公司在估计期的平均报酬率，之后将事件窗口的报酬率减去该平均报酬率就是平均调整后的异常报酬率。合并市场模型系数值与过去平均报酬率的语句如表 16-7 所示。

表 16-7

```
/*将估计期的市场模型以及个股过去的平均报酬率计算出来*/
```

```
data a;
merge b c;by event stked;
if _p+_edf_=&est then output;
run;
data newevent;
merge event a;by event stked;
if mean=. then delete;
run;
```

由表 16-7 可知，将平均报酬率与市场模型的系数值合并，以方便之后直接计算三种异常报酬率。接下来，使用 proc sql 就可以将三种异常报酬率直接计算出来。抓取事件窗口三种异常报酬率数值的语句如表 16-8 所示。

表 16-8

```
PROC SQL;
CREATE table b
as select
a.stkcd,
a.event,
b.t-a.t as t,
b.ret-a.intercept-b.rm*a.rm as AR1,
/*
可在此处修改成 3 因子、4 因子、5 因子建构下的异常报酬率
*/
b.ret-a.mean as AR2,
b.ret-b.rm as AR3
from newevent a, ret b
where (a.stkcd=b.stkcd
and a.t-&before <=b.t <= a.t+&after);quit;
/*至此 直接算出三种异常报酬率
AR1 市场模型的异常报酬率
AR2 平均调整后的异常报酬率
AR3 市场调整后的异常报酬率*/
```

由表 16-8 所示的语法就可以直接计算出需要的三种异常报酬率的数据，接下来就可以计算累积异常报酬率，以及检定异常报酬率的显著性，计算窗口期间的异常报酬率显著性的语句如表 16-9 所示。

表 16-9

```
proc univariate data=b ;  
var AR1-AR3;  
by t;  
ods output TestsForLocation=test BasicMeasures=basic;  
run;
```

在如表 16-9 所示的程序中，先检定每一天的异常报酬率是否有显著，利用 proc univariate 的语法，可以协助研究者完成该部分的检定。整理异常报酬率检定表格的语句如表 16-10 所示。

表 16-10

```
data test;  
set test;  
if 1<=mod(_n_,3)<=2 then output;  
drop testlab ptype mu0 test;  
run;  
data test;  
merge basic test;  
if pvalue<0.01 then value=round(locvalue,0.001)||'***';  
else if pvalue<0.05 then value=round(locvalue,0.001)||'**';  
else if pvalue<0.1 then value=round(locvalue,0.001)||'*';  
else value=round(locvalue,0.001)||'';  
run;  
proc sort data=test;by varname model t;  
run;  
proc transpose data=test out=test(drop=varname _name_);  
var value;  
id locmeasure;  
by varname model t;  
run;
```

运行如表 16-10 所示程序，仔细将数据进行整理之后，检定每日异常报酬率的结果如图 16-2 所示，就可以将表格轻松地呈现了，接下来就可以描绘出累积异常报酬率的走势图。计算累积异常报酬率的语句如表 16-11 所示。

	model	t	Mean	Median
1	Market Model	-2	0.066	-0.074
2	Market Model	-1	0.09	-0.082
3	Market Model	0	-0.005	-0.108
4	Market Model	1	0.035	-0.067
5	Market Model	2	-0.018	-0.204***
6	Mean Adjusted Model	-2	-0.118	-0.253***
7	Mean Adjusted Model	-1	-0.103	-0.128***
8	Mean Adjusted Model	0	-0.161**	-0.136***
9	Mean Adjusted Model	1	-0.216***	-0.177***
10	Mean Adjusted Model	2	-0.187***	-0.245***
11	Market Adjusted Model	-2	0.096	-0.06
12	Market Adjusted Model	-1	0.121*	-0.06
13	Market Adjusted Model	0	0.04	-0.11**
14	Market Adjusted Model	1	0.089	-0.035
15	Market Adjusted Model	2	0.027	-0.16***

图 16-2

表 16-11

```

proc sort data=basic;by model t;
run;
data basic;
set basic;
if mod(_n_,2)=1 then output;
run;
data basic;
set basic;
retain CAR 0;
CAR=CAR+locvalue;
if model^=lag(model) then CAR=Locvalue;
run;

```

由于计算累积异常报酬率，就是进行平均异常报酬率的累加，所以可以采用上述语法，将异常报酬率的均值取出，进一步计算不同类型的异常报酬率。绘制累积异常报酬率走势图的语法如表 16-12 所示。

表 16-12

```

proc sgplot data=basic;
title "The CAR during the event date";
series x=t y=CAR/ group=model;
run;

```

使用该语法，就可以绘制出如图 16-3 所示的异常报酬率的走势图。

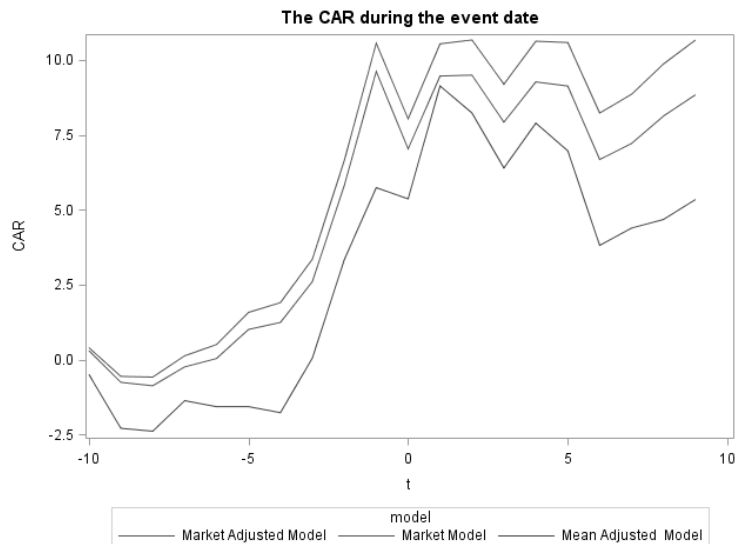


图 16-3

最后将异常报酬率的数据与异常报酬率检定的数据——输出即可，输出异常报酬率相关表格的语法如表 16-13 所示。

表 16-13

<pre>proc export data=b outfile="D:\The Application of SAS in Financial Research\CH16\output\data_ar.csv" replace; run; proc export data=test outfile="D:\The Application of SAS in Financial Research\CH16\output\test_ar.csv" replace; run; /* 采用 CSV 存盘较佳，因为异常报酬率数据可能会超过 65535 笔 当然，保留为 SAS 的文档会更好 */</pre>
--

16.2 日历期间投资组合

虽然早期的财务研究都着重于短期的市场反应，但仍有不少研究探讨各个事件的长期绩效，Fama

(1998) 针对长期的事件研究法进行了深入的探讨, 在长期的事件中如果采用累积异常报酬率, 其统计检定力较差, 因此多数研究采用买进持有异常报酬率以及日历期间投资组合法来探讨事件的长期异常绩效, 本节首先介绍日历期间投资组合法, 16.3 节则介绍买进持有异常报酬率法。

在日历期间投资组合法中, 第一步, 要对每个月形成事件投资组合, 其做法类似于移动窗口的方式, 在 t 月时, 针对 $t-1$ 到 $t-T$ 月曾经发生过事件的公司形成投资组合, 假设 T 为 36, 则如果目前为 2000 年 1 月, 若 A 公司在 1997 年 1 月到 1999 年 12 月间曾经发生过我们所要求的事件, 就纳入为事件投资组合; 若另外有一家 B 公司是在 1996 年 12 月发生该事件, 则由于已经超过 36, 在 2000 年 1 月时, 事件投资组合不会纳入该公司, 但是在 1997 年 1 月到 1999 年 12 月间, 该公司都会被纳入到事件投资组合中。

第二步, 便是计算样本期间内所有事件投资组合每个月份的等值加权平均报酬率, 以及市值加权平均报酬率。

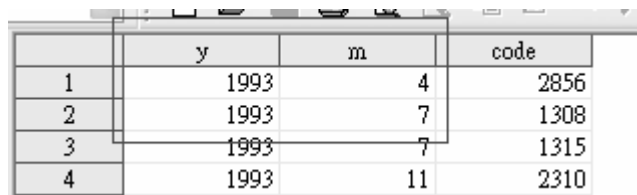
最后, 便是计算该投资组合的 3 因子、4 因子模型, 以估计出的截距项作为长期异常报酬率的绩效, 以下以 example_16_2.sas 为例来介绍日历期间投资组合的方法。整理长期事件研究法的语句如表 16-14 所示。

表 16-14

```

libname aa 'D:\The Application of SAS in Financial Research\CH16\data';
data a;
set aa.long;
run;
proc sort data=a;by y m;
run;
```

表 16-4 的开始, 程序先读取事件的月份, 在本范例中, 选取的事件为更换财务主管的事件月份。事件研究法的部分数据如图 16-4 所示。



	y	m	code
1	1993	4	2856
2	1993	7	1308
3	1993	7	1315
4	1993	11	2310

图 16-4

确定时间最早开始于 1993 年, 研究者就可以准备时间变量的数据, 数据合并之后就可以确定各个事件发生的 t 月份。针对样本期间的月份进行编码的程序如表 16-15 所示。

样本期间编码的结果如图 16-5 所示。

表 16-15

```
data t;  
do y=1993 to 2006;  
do m=1 to 12;  
output;  
end;  
end;  
run;  
data t;  
set t;  
t=_n_;  
run;  
data a;  
merge a t;by y m;  
if code=. then delete;  
run;  
proc sort data=a;by code;  
run;
```

	y	m	code	t
1	1995	1	1101	25
2	2002	1	1101	109
3	2003	7	1101	127
4	2005	3	1101	147
5	2006	2	1101	158
6	1998	4	1102	64
7	2000	8	1102	92

图 16-5

接下来，针对股票报酬率进行处理，将每个月份的报酬率也标上 t，以便在之后计算长期报酬率，其中由于需要计算市值加权报酬率，所以仿照动能投资组合的做法，也求得前一期的市值。合并样本报酬率与期间编码的程序如表 16-16 所示。

表 16-16

```
libname aa 'D:\SAS 在财务研究上的运用\SAS data\月股价';  
data ret;  
set aa.price;  
keep y m code mv ret;  
run;
```

续表

```

proc sort data=ret;by code y m;
run;
/*要计算等值加权与市值加权 所以先抓出前一个月的市值*/
data ret;
set ret;
mv=lag(mv);
if code^=lag(code) then mv=.;
run;
proc sort data=ret;by y m;
run;
data ret;
merge ret t;by y m;
if t=. then delete;
run;

```

接着利用事件月份以及股票报酬率的数据,找出每个月的事件投资组合,由于投资组合是由每 t 月针对第 $t-1$ 月到第 $t-36$ 月曾经发生事件的公司形成的,可以反过来思考,当一家公司在第 $t-36$ 月时发生事件,则第 $t-35$ 月到 t 月都会纳入投资组合;亦即,如果一家公司在第 t 月发生事件,则第 $t+1$ 月到第 $t+T$ 月都要纳入事件投资组合。采用该逻辑,便可以撰写出如表 16-17 所示的抓取日历期间报酬率数值的程序语法。

表 16-17

```

%macro ctp(t);
proc sql;
create table event&t
as select
b.y as y,
b.m as m,
b.code as code,
b.mv as mv,
b.ret as ret
from a a, ret b
where a.t+1<=b.t<=a.t+&t
and a.code=b.code;
quit;

```

撰写 ctp 的宏时,指定 t 变量,研究者可以根据需要,采用过去 12 个月或者 36 个月进行研究。利用 proc sql 的语法,针对事件月份抓出股票报酬率,仅需要在条件语句中加入 $a.t+1 \leq b.t \leq a.t+\&t$ 即可,

如此一来，SAS 就会自动抓取各个时点下符合条件的股票了。

某些公司可能会重复地发生某个特定事件，如果 A 公司在第 t 月以及第 t+1 月连续发生特定事件，那么根据 proc sql 的语法，其从第 t+2 月到第 t+T 月的月份中，A 公司的数据会重复出现，虽然这样符合 t 月份发生某一事件时，要抓取其 t+1 月到 t+T 月的报酬率，但是，本程序真正要处理的是在 t 月份时的报酬率，如果该家公司在 t-1 到 t+T 曾经发生过事件，就投资组合而言，即便其过去曾经发生了四次事件，但在 t 时点仍然只采集一次样本，因此需要利用 proc sort 的语法删除重复的数据，删除重复报酬率数据的语法如表 16-18 所示。

表 16-18

<pre>proc sort data=event&t nodupkey;by y m code; run;</pre>
--

计算日历期间均权与市值加权报酬率的语法如表 16-19 所示。

表 16-19

<pre>proc means noprint data=event&t; var ret; weight mv; by y m; output out=vw(drop=_type__freq_) mean=vwret; run; proc means noprint data=event&t; var ret; by y m; output out=ew(drop=_type__freq_) mean=ewret; run; data event&t; merge vw ew;by y m; run;</pre>
--

在该部分，表 16-19 所示的程序是分别计算事件投资组合的市值加权以及等值加权报酬率，并且在最后合并，在整理完事件投资组合的报酬率之后，就可以进行 3 因子以及 4 因子模型的估计了。合并样本日历期间报酬率与因子报酬率数据的程序如表 16-20 所示。

表 16-20

<pre>libname aa 'D:\The Application of SAS in Financial Research\SAS data\four factor'; data event&t; merge event&t aa.factor;by y m; vw=vwret-rf;</pre>
--

续表

```
ew=ewret-rf;
rmrf=rm-rf;
if vw=, or ew=, then delete;
run;
```

在如表 16-20 所示的语法中，将事件投资组合的报酬率与因子数据进行合并，在宏语法的最后部分，利用第 11 章的回归宏，便可以顺利地将结果直接完成。计算日历期间报酬率的三种因子模型回归的程序如表 16-21 所示。

表 16-21

```
proc reg data=event&t;
model ew=rmrf smb hml mom/white ;
model ew=rmrf /white;
model ew=rmrf smb hml/white;
model ew=rmrf smb hml mom/white;
model vw=rmrf /white;
model vw=rmrf smb hml/white;
model vw=rmrf smb hml mom/white;
ods output FitStatistics=fit;
ods output NObs=nobs;
ods output ParameterEstimates=para;
quit;
%include 'D:\The Application of SAS in Financial Research\CH11\program_cn\reg.sas';
%ools(3);
```

利用第 11 章所撰写的回归宏¹，可以直接得到日历时间投资组合的 CAPM、3 因子以及 4 因子的结果，接着就是利用 proc export 将得到的回归结果输出到 Excel 文档中，输出日历期间报酬率回归图的结果如表 16-22 所示。

表 16-22

```
proc export data=ols
outfile="D:\The Application of SAS in Financial Research\CH16\output\CTP"
dbms=xlsx
replace;
```

1 在本节中，我们将呼叫宏的语法包含在另外的宏之中，会有相当大的风险，这是由于不同的宏可能会使用相同的指定变量名，若要尝试在宏 A 中呼叫宏 B，切记宏 A 中的指定变量越简单越不常用越好，在本 %ctp 中使用指定变量 t 是本书中第一次使用，就是为了避免不同宏之间指定变量的相互影响。

续表

```
sheet="&T";
quit;
%mend;
%ctp(12);
%ctp(36);
```

在输出到 CTP 的 Excel 文档中之后,就可以由如图 16-6 所示的日历期间因子报酬率回归结果,检查该事件是否有长期的异常绩效了。

A	B	C	D	E	F	G	H
Variable	MODEL2	MODEL3	MODEL4	MODEL5	MODEL6	MODEL7	tvalue
Intercept	0.091 (0.23)	-0.225 (-0.94)	-0.203 (-0.84)	0.085 (0.31)	-0.033 (-0.13)	0.02 (0.08)	OLS
rmrf	0.884* (18.73)	0.978* (31.84)	0.976* (31.6)	0.887* (26.62)	0.926* (28.41)	0.921* (28.32)	OLS
SMB		0.68* (13.46)	0.662*** (11.84)		0.264* (4.92)	0.221* (3.74)	OLS
HML		0.195* (6.31)	0.191*** (6.02)		0.051 (1.55)	0.039 (1.18)	OLS
MOM			-0.03 (-0.74)			-0.074* (-1.71)	OLS
Adj R-Sq	68.62%	88.3%	88.26%	81.56%	84.42%	84.62%	OLS
R-Square	68.82%	88.52%	88.56%	81.68%	84.72%	85%	OLS
N	161	161	161	161	161	161	OLS

图 16-6

由图 16-6 可见, T=12 的结果, 由 MODEL2 到 MODEL4 为均等加权平均报酬的回归结果, 由 MODEL5 到 MODEL7 为市值加权平均报酬率的结果, 结果发现没有长期的异常报酬率现象, 即使是采用 T=36 亦没有显著的异常报酬率, 在 16.3 节中笔者将进一步介绍采用买进持有异常报酬率法来检验更换财务主管是否具有异常绩效。

16.3 买进持有异常报酬率：配对投资组合法

买进持有异常报酬率是根据各家公司事前的特征值, 寻求其相对应的标杆投资组合 (benchmark portfolio), 并且计算事件公司与标杆投资组合的长期绩效之后, 进行两个投资组合的差异检定。标杆投资组合的选择有两种方法, 第一种为配对投资组合法, 第二种为配对样本法, 本节先介绍配对投资组合法, 接着再介绍配对样本法。

实际上, 配对投资组合多以 Daniel, Grinblatt, Titman, and Wermers (DGTW, 1997) 的方法来进行分组的。大部分的文献都是这样叙述其分组方式的, 先是将各只股票依照其市值分成 5 个投资组合, 再将每个规模投资组合分出 5 个 BM 投资组合, 最后再将这 25 个投资组合依照其过去的报酬率分 5 个动

能投资组合，得到 125 个投资组合。在解释上虽然清楚，但是在撰写程序上，仍然有所不足，分出投资组合的频率是多少？将详细了解清楚每月、每季或者每年的情况方能够写出适合的程序语法。本质上，DGTW 的做法和 Fama and French (1993) 相似，在 t 年 6 月底形成投资组合，首先要求公司拥有 t 年 6 月底的市值数据以及 $t-1$ 年 12 月底的 BM 数据，并且各家公司此时需要存在 2 年以上，而在动能投资组合时，计算其 $t-1$ 年 7 月到 t 年 5 月的报酬率，至少要有 6 个月的报酬率数据，最后根据这些数据依次做出五乘五乘五的投资组合，之后到 $t+1$ 年 6 月底重新分组，每只股票所属的投资组合都应维持 1 年。

为了介绍程序的简便，在本节中仅采用公司在 t 年 6 月底有市值数据以及 $t-1$ 年 12 月底有 BM 数据的条件，为了简便起见，BM 采用原始的 price to book ratio¹，并且不要求公司至少要存在 2 年以上，采用 example_16_2.sas 来介绍市值与市净比整理的语法如表 16-23 所示。

表 16-23

```

DMLOG;CLEAR;OUT;CLEAR';
option nonotes nlabel;
libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price';
data mv;
set aa.price;
if m=6 then output;
keep y   code mv;
run;
proc sort data=mv;by y ;
run;
data pb;
set aa.price;
if m=12 then output;
keep y   code pb;
run;

```

在如表 16-23 所示的程序一开始，先分别取出每年 6 月的市值数据以及每年 12 月的 PB 数据，如同先前所述，在进行分组之时，是利用 $t-1$ 年 12 月底的 PB 与 t 年 6 月底的市值进行分组的，因此再要对 PB 进行处理，PB 处理的语法如表 16-24 所示。

表 16-24

```

proc sort data=pb;by code y ;
run;
data pb;

```

1 Fama and French (1993) 的权益账面价值会加回递延贷项、递延所得税，并且扣除特别股的股利的数值，而在 DGTW 中，BM 是采用产业调整后的信息，而不是使用公司的原始数据。

续表

```
set pb;  
pb=lag(pb);  
if code^=lag(code) then delete;  
run;  
proc sort data=pb;by y;  
run
```

将 PB 处理完毕之后，就可以将股票数据进行合并的动作，最后就可以顺利分出标杆投资组合了。计算每档股票每年的分组语法如表 16-25 所示。

表 16-25

```
data bench;  
merge mv pb;by y ;  
if mv=. or pb=. then delete;  
run;  
proc rank data=bench out=bench groups=5;  
var mv;  
by y;  
ranks mvrnk;  
run;  
proc sort data=bench;by y mvrnk;  
run;  
proc rank data=bench out=bench groups=5;  
var pb;  
by y mvrnk;  
ranks mbrnk;  
run;
```

由表 16-25 的程序可知，在合并完后，先将市值或者 PB 数据有缺失的公司删除，再将公司依照市值分成 5 组，由于要进行 dependent sort，再将公司依照年度、市值组合排序，就可以在每个市值投资组合里再分出 5 个 MB 投资组合了。将分组变量扩展为 1 到 12 月份的语法如表 16-26 所示。

表 16-26

```
data bench;  
set bench;  
mvrnk=mvrnk+1;  
mbrnk=mbrnk+1;  
drop mv pb;
```

续表

```
do m=1 to 12;
output;
end;
run;
```

在如表 16-26 的程序中，采用了一个小诀窍，由于分组是每年分类一次，因此需要将其延伸为 1 年期间，而由于 SAS 分组为由 0 开始，此处也将其调整为由 1 开始，但是在这个语法之后，会产生一个问题，数据展开后会产生如图 16-7 所示的月份扩展结果的情况。

	y	code	mvrank	mbrank	m
1	1981	1402	1	2	1
2	1981	1402	1	2	2
3	1981	1402	1	2	3
4	1981	1402	1	2	4
5	1981	1402	1	2	5
6	1981	1402	1	2	6
7	1981	1402	1	2	7
8	1981	1402	1	2	8
9	1981	1402	1	2	9
10	1981	1402	1	2	10
11	1981	1402	1	2	11
12	1981	1402	1	2	12

图 16-7

如图 16-7 所示，采用月份展开后，确实会有一年的数据，但是程序语法的目标是以 t 年 6 月底分组的结果作为 t 年 7 月到 t+1 年 6 月的组合，会发现在 m=1 到 6 的情况下，该语法皆不能顺利进行，因此需要做进一步的调整。在年度整理中合并报酬率适合年份的语句如表 16-27 所示。

表 16-27

```
data bench;
set bench;
if m<=6 then y=y+1;
run;
```

此处加入了这样的条件，if m<=6 then y=y+1，就可以将分组进行修正，程序执行之后，就是完整的分组结果了，整理后可合并的文档如图 16-8 所示。

	y	code	mvrank	mbrank	m
1	1982	1402	1	2	1
2	1982	1402	1	2	2
3	1982	1402	1	2	3
4	1982	1402	1	2	4
5	1982	1402	1	2	5
6	1982	1402	1	2	6
7	1981	1402	1	2	7
8	1981	1402	1	2	8
9	1981	1402	1	2	9
10	1981	1402	1	2	10
11	1981	1402	1	2	11
12	1981	1402	1	2	12

图 16-8

接着将分组信息与股票报酬率合并，并且计算标杆投资组合的每个月的报酬率。整理报酬率与市值数据的语句如表 16-28 所示。

表 16-28

```

data ret;
set aa.price;
ret=1+0.01*ret;
keep code y m mv ret;
run;
proc sort data=ret;by code y m mv ret;
run;
data ret;
set ret;
mv=lag(mv);
if code^=lag(code) then delete;
run;

```

如表 16-28 所示，在报酬率的数据中，由于之后要计算买进持有报酬率，因此需要用到几何平均报酬率的做法，在此处先将报酬率化为 $1+Ret$ 的形式，方便进行之后的运算。接着，取出各只股票前一个月的市值数据，以作为市值加权报酬率的加权值，在此处要特别小心，所使用的市值一定是这只股票前一个月月底的市值数据，而不是同一个月份的市值数据。求算标杆投资组合市值加权报酬率的语法如表 16-29 所示。

表 16-29

```

data final;
merge bench ret;by code y m;
if mvrnk=., or mbrnk=., or ret=., then delete;
run;
proc sort data=final;by y m mvrnk mbrnk;
run;
proc means data=final noprint;
var ret;
weight mv;
by y m mvrnk mbrnk;
output out=benchret(drop=_type_ _freq_) mean=benchret;
run;

```

计算出市值加权平均报酬率后，就可以分别计算个别公司与事件公司的长期持有报酬率，为了之后语法的方便使用，将标杆投资组合的两个分组变量合并为 1 个变量，将标杆投资组合变量合并的语法如表 16-30 所示。

表 16-30

```

data benchret;
set benchret;
bench=mvrank||mbrank;
/*如果有超过 2 种分组组合，亦可采用相同的方式 如 new=a||b||c; 就是结合 3 变量为 1*/
run;

```

利用合并变量的语法之后，便可以使用宏语法来进行计算，长期买进持有报酬率的宏语法如表 16-31 所示。

表 16-31

```

%macro h(file,code,time,ret,p,h);
  proc sort data=&file(keep=&time) out=time nodupkey;by &time;
  run;
  data time;
    set time;
    time=_n_;
  run;
  proc sort data=&file;by &time;
  run;
  data &file;
    merge &file time;by &time;
  run;
  proc sort data=&file;by &code descending time;
  /*
  revise 20150914
  因为时间需要降幂排序 如果有多个变量 无法使用多个 descending
  */
  run;
  data &file;
    set &file;
    %do i=1 %to &h;
      r&i=lag&i(&ret);
      &p&i=(geomean(of r1-r&i)**n(of r1-r&i));
      if &code^=lag&i(&code) then &p&i=.;
    %end;
    drop r1-r&h time;
  run;
%mend;
%h(benchret, bench, y m, benchret, bhr,36);
%h(final,code,y m,ret,idiret,36);

```

表 16-31 的语法指定了 6 个变量：file 为文档来源，code 为指定股票代码，time 为指定时间变量，ret 为报酬率变量，p 为指定计算买进持有报酬率变量，h 为指定持有该资产的期数。呼叫宏程序执行后，就可以顺利地将报酬率计算完毕。接下来可以读取事件数据，进行后续检定。求得事件样本的买进持有报酬率的语法如表 16-32 所示。

表 16-32

libname aa 'D:\The Application of SAS in Financial Research\CH16\data';
data event;
set aa.long;
event=1;
run ;
proc sort data=event;by code y m;
run ;
data event;
merge event final;by code y m;
if event=1 then output;
run ;
proc sort ;by mvrank mbrank y m;
run ;
data bhar;
merge event benchret;by mvrank mbrank y m;
if event=1 then output;
run ;

在表 16-32 的语法中，合并了事件公司的报酬率与其标竿投资组合的报酬率数据，之后便可以分别计算其买进持有异常报酬率的数据。

使用如表 16-33 所示的 BHAR 宏程序，可以计算出每一期的买进持有异常报酬率的数据，之后便可以进行所需要的检定。

表 16-33

%macro bhar(h);
data bhartest;
set bhar;
%do i=1 %to &h;
BHAR&i=(idiret&i-bhr&i)*100;
%end ;
drop idiret1-idiret&h bhr1-bhr&h;
run ;
%mend ;

检定 1、2、3 年买进持有报酬率的语法如表 16-34 所示。

表 16-34

```
proc univariate data=bhartest ;
var bhar12 bhar24 bhar36;
ods output TestsForLocation=test
BasicMeasures =moment;
run;
data test;
set test;
if mod(_n_,3)=0 then delete;
if pvalue<0.01 then sig='***';
if 0.01<=pvalue<0.05 then sig='**';
if 0.05<=pvalue<0.1 then sig='*';
keep varname sig;
run;
data moment;
set moment;
stat=round(LocValue,0.001);
if 0<mod(_n_,4)<3 then output;
keep varname locmeasure stat;
run;
data final;
merge moment test;
run;
```

长期买进持有异常报酬率检定结果的表格如图 16-9 所示，虽然事件公司买进异常报酬率的均值并未显著不等于 0，然而其中位数却显著不等于 0，显示更换财务主管的长期绩效会较配对投资组合差。但是由于采用配对投资组合，可能会由于许多事件公司都属于同一种风格，故难以确定该差异是否是由于投资组合的绩效过差而造成的，因此需要进行配对样本的检定。

	VarName	LocMeasure	stat	sig
1	BHAR12	均值	-1.388	
2	BHAR12	中位数	-7.078	***
3	BHAR24	均值	-5.588	***
4	BHAR24	中位数	-15.175	***
5	BHAR36	均值	-4.565	*
6	BHAR36	中位数	-15.158	***

图 16-9

16.4 买进持有异常报酬率：配对样本法

在 16.3 节中，采用配对投资组合法作为标杆投资组合的计算，而在本节的配对样本中，则是采用 1 对 1 的控制样本作为事件公司的标杆投资组合。在进行配对样本时通常有两种方式：一种是要求配对公司不能在过去数年内曾经发生过该事件，然而却很有可能在配到该公司之后，该配对公司亦发生了该事件；另外一种方式则是要求在测试期间内，控制公司不能发生该事件。第一种配对方式比较符合真实情境，但不可避免地会发生先发生事件的公司晚期绩效减去后发生事件的早期绩效的事件重迭期，但却有较佳的投资意向，第二种配对方式则能够清楚地观察到纯粹由事件所驱动的异常报酬率，虽然投资意向不如第一种方式好，但是能够更精准地描述异常事件的绩效，接下来我们以 example_16_4.sas 为例来介绍如表 16-35 所示的求得样本的市值比与市值的语法。

表 16-35

```
option nonotes nlabel;
libname aa 'D:\The Application of SAS in Financial Research\SAS data\monthly price';
data mv;
set aa.price;
if m=6 then output;
keep y   code mv;
run;
proc sort data=mv;by y ;
run;
data pb;
set aa.price;
if m=12 then output;
keep y   code pb;
run;
proc sort data=pb;by code y ;
run;
data pb;
set pb;
pb=lag(pb);
if code^=lag(code) then delete;
run;
proc sort data=pb;by y;
run;
data bench;
merge mv pb;by y ;
if mv=. or pb=. then delete;
```


续表

```

run;
data bench;
set bench;
do m=1 to 12;
output;
end;
run;
data bench;
set bench;
if m<=6 then y=y+1;
run;
proc sort data=bench;by code y m;
run;

```

在表 16-35 所示的程序中我们先准备市值以及 PB 的数据，参照配对投资组合的方式，在选取配对公司时，采用其前一年的 PB 和市值来进行配对。特别要注意的是，在进行配对时一定要用事前的数据。取得样本前一期的报酬率的语法如表 16-36 所示。

表 16-36

```

data ret;
set aa.price;
keep code y m mv ret;
run;
proc sort data=ret;by code y m;
run;
data ret;
set ret;by code;
weight=lag(mv);
if first.code then delete;
drop mv;
run;
data bench;
merge bench ret;by code y m;
run;
proc sort data=bench;by y m;
run;
proc sort data=ret nodupkey out=time(keep=y m);by y m;

```

续表

```
run;
data time;
set time;
t=_n_;
run;
data bench;
merge bench time;by y m;
run;
proc sort data=bench;by code y m;
run;
```

在表 16-36 所示的程序中，是将数据与股票报酬率进行合并，并且同时产生时间变量，以便之后得以方便抓取 T 个月的报酬率。将样本区分为事件样本与控制样本的语法如表 16-37 所示。

表 16-37

```
libname aa 'D:\The Application of SAS in Financial Research\CH16\data';
data event;
set aa.long;
event=1;
run;
proc sort data=event nodupkey out=code (keep=code event);by code;
run;
data control;
merge code bench;by code ;
if event=1 then delete;
if mv=. or pb=. then delete;
drop event;
run;
data event;
merge event bench;by code y m;
if mv=. or pb=. then delete;
if event=1 then output;
drop event;
run;
```

如表 16-37 所示的语法为创造最干净的控制样本，亦即在所有的样本期间内都不曾发生过更换财务主管的公司，在后续的研究中才会作为其配对样本的数据，若曾经发生过更换财务主管的事件，则会立即删除数据，或者读者亦可计算其过去三年是否发生过该事件，如果没有发生，则可以纳入为样

本。求出符合条件的配对样本的语法如表 16-38 所示。

表 16-38

```

proc sql;
create table final
as select
a.code as event_code,
a.event as event,
b.code as control_code,
a.t as t
from event a,control b
where
a.t=b.t and
a.mv*(1-&percent)<=b.mv<=a.mv*(1+&percent) and
a.pb*(1-&percent)<=b.pb<=a.pb*(1+&percent);
quit;
proc sort data=final;by event;
run;

```

在表 16-38 所示的语法中，我们利用 proc sql 的语法，挑选出配对公司的市值与 PB 比率介于事件公司的上下个特定比率的公司，在本例中，我们将其预设为 30%，事实上当条件越设越多时，其上下比率的幅度可能就要更高，否则可能会配不到适合的公司，为了简便起见，在本例中，有一个重要的条件并没有设立，那就是配对公司要跟事件公司为同一个产业的条件。在做配对样本时设定该产业相同的条件较佳，但为了增加研究样本，该程序中省略了该步骤。随机挑选出每个事件样本的一家公司配对样本的语法如表 16-39 所示。

表 16-39

```

proc sort data=final out=final1(keep=event_code event t) nodupkey;by event;
run;
proc surveyselect data=final
out=final2(keep=control_code event t rep) noprint
seed=1
n=1
rep=1000
method=srs;
strata event;
run;

```

在表 16-39 所示的程序中，采用随机抽样的语法抽出特定的配对样本，因为同一家公司其符合条

件的配对公司可能有很多家，因此采用随机抽样的方式，在每个事件里随机抽出一家公司作为其配对公司，但有部分学者认为：采用一次配对样本计算绩效来检验绩效其检定力可能不够，因此建议采用拔靴法进行检定，即将 rep 设定为 1000，利用该 1000 个配对样本的组合所估计出来的绩效分配来评判异常报酬率是否显著较佳或较差。计算配对样本报酬率的数据如表 16-40 所示。

表 16-40

```

%macro Bhar(t);
proc sql;
create table event_ret
as select
a.event as event,
b.ret/100+1 as ret,
b.weight as mv,
b.t-a.t as t
from final1 a,bench b
where
a.t+1<=b.t<=a.t+&t and
a.event_code=b.code;
quit;
proc sort data=event_ret;by t;
proc sql;
create table control_ret
as select
a.event as event,
a.replicate as replicate,
b.ret/100+1 as ret,
b.weight as mv,
b.t-a.t as t
from final2 a,bench b
where
a.t+1<=b.t<=a.t+&t and
a.control_code=b.code;
quit;
proc sort data=control_ret;by replicate t;
run;
%mend;
%Bhar(36);

```

在如表 16-40 所示的程序语法中，分别抓取事件公司以及配对公司发生事件日后 1 到 T 个月的报酬率，接着便可以计算其投资组合的绩效。计算样本与配对样本的平均报酬率的语法如表 16-41 所示。

表 16-41

```

proc means data=event_ret noprint;
var ret;
by t;
output out=event(keep=ret t) mean=ret;
run;
proc transpose data=event out=event;
var ret;
run;
proc means data=control_ret noprint;
var ret;
weight mv;
by replicate t;
output out=control(keep=replicate t ret) mean=ret;
run;
proc transpose data=control out=control;
var ret;
by replicate;
run;

```

表 16-41 所示的语法为整理事件公司投资组合绩效以及 1000 次配对公司投资组合的绩效，接着便可以计算其买进持有期报酬率，计算持有期报酬率的程序如表 16-42 所示。

表 16-42

```

%macro hold_return(file,h);
data &file;
set &file;
%do i=1 %to &h;
hold&i=(geomean(of col1-col&i)**n(of col1-col&i)-1)*100;
%end;
keep hold1-hold&h;
run;
%mend;
%hold_return(event,36);
%hold_return(control,36);

```

如表 16-42 所示的宏语法已经重复出现过数次，其为分别计算买进持有期报酬率的语法，但是读者要仔细检验 event 以及 control 的文档型态，事实上由于经由转置语法的处理，T 期的报酬率会转为 col1 到 colT 的格式，这在撰写宏语法时有相当大的帮助。

分别处理完事件公司报酬率以及配对样本公司报酬率之后，要找出事件公司报酬率是否是在 1000 次配对样本公司报酬率分配的那个百分比之处，以便决定其报酬率是显著高于或者显著低于配对样本的报酬率。将配对样本持有期报酬率进行转置的语法如表 16-43 所示。

表 16-43

<pre>proc transpose data=control out=a(drop=_name_); var hold1-hold36; run; data a; set a; t=_n_; run; proc transpose data=a out=a; var col1-col1000; by t; run;</pre>
--

由表 16-43 可知，利用重复转置的语法，可以将 36 期的异常报酬率全部放在同一个列上，如此能够方便程序的撰写。计算配对样本报酬率分布的语法如表 16-44 所示。

表 16-44

<pre>proc univariate data=a noprint; var col1; by t; output out=Pctls pctlpts = 0.5 2.5 5 95 97.5 99.5 pctlpre = p; output out=ref mean=reference; run;</pre>

由表 16-44 所示的程序可见，利用 proc univariate 的语法，可以求得各个变量的任意百分位数，其中 pctlpts 指令是要求输出各个百分位数的数值，由于可以一次输出不同的百分位数，因此 pctlpre 为百分位数变量前缀的语法，我们用 p 作为前缀，在先前的部分之所以会将数据转置成为 1 列数据，就是为了在此处便利使用，同时求出这 1000 次拔靴法的配对样本所组成的配对组合的平均报酬率，SAS 输出的结果如图 16-10 所示。

	t	p0_5	p2_5	p5	p95	p97_5	p99_5
1	1	-1.255911104	-1.128979543	-1.018287748	-0.027489083	0.0837995574	0.2082290411
2	2	-1.928133542	-1.655128942	-1.520864306	0.0136124839	0.162415652	0.3743067657
3	3	-4.193248982	-3.98159503	-3.800902612	-1.787699198	-1.565381021	-1.137671306
4	4	-1.93776699	-1.683237655	-1.539391605	0.9766541391	1.2443185774	1.7751779385
5	5	-4.205880944	-3.866057664	-3.701227796	-1.063600999	-0.697763519	-0.000690714
6	6	-3.056895335	-2.677323131	-2.451180108	0.4352656156	0.6782942158	1.1691014503
7	7	-0.484708826	-0.157899787	0.0732178544	2.5558031896	2.8175139347	3.2856776659
8	8	1.7814428468	2.1219610039	2.3804244157	5.6894679173	5.9821185896	6.5100742117
9	9	2.5919066034	3.107891779	3.4007372509	7.2951363399	7.6552641083	8.3484336048
10	10	1.9205688749	2.4104743614	2.6997258086	7.5340984314	7.8860684153	8.2883789191
11	11	2.4373103635	2.8471807687	3.146777092	7.7404000884	7.997011625	8.7005343583
12	12	5.9411628224	6.4307349449	6.7537751419	11.119927098	11.417012494	12.222934362
13	13	9.3489688049	9.9095631928	10.221319751	14.99435289	15.364554865	16.137154796
14	14	10.057869568	10.571588467	10.913138814	15.427628603	15.762923494	16.503712469
15	15	16.338862996	16.90078603	17.288279873	21.811305833	22.167639943	22.948975598
16	16	9.8709672357	10.464677148	10.755098768	15.26691272	15.548380217	16.237024088
17	17	16.440860003	17.256661104	17.653766676	23.008589806	23.368378535	24.132320865

图 16-10

SAS 将小数点以_取代, p0_5 就是百分位数在 0.5 的位置, 该表格创建出来之后, 便可以一一检定事件公司是否有显著的正或者负异常报酬率绩效。事件样本报酬率转置的语句如表 16-45 所示。

表 16-45

```
proc transpose data=event out=b;
run;
```

如表 16-45 所示的语法依然将数据进行转置, 如此一来就可以直接跟百分位数表格合并, 只要买进持有期报酬率低于 5、2.5 甚至 0.5 百分位数的绩效, 读者就可以宣称事件公司有负的异常买进持有期报酬率; 反之买进持有期报酬率若高于 95、97.5 或 99.5 百分位数的绩效, 就可以宣称事件公司有正的买进持有期异常报酬率。求事件样本与配对样本报酬率均值差异的语法如表 16-46 所示。

表 16-46

```
data final;
merge b ref;
event_return=round(col1,0.001);
ref_return=round(reference,0.001);
dif_return=event_return-ref_return;
keep _name_ event_return ref_return dif_return;
run;
```

在如表 16-46 所示的程序中, 我们先将事件投资组合的绩效与对照投资组合的平均绩效进行合并, 接着计算其差异, 便可以简单地看出长期异常绩效的状况。配对样本报酬检定的语法如表 16-47 所示。

表 16-47

```
data final;
merge final pctl5;
drop t;
```

续表

```
if event_return<P0_5 then do;
sign='-';sig='***';
end;
else if event_return<P2_5 then do;
sign='-';sig='**';
end;
else if event_return<P5 then do;
sign='-';sig='*';
end;
else if event_return>P99_5 then do;
sign='+';sig='***';
end;
else if event_return>P97_5 then do;
sign='+';sig='**';
end;
else if event_return>P95 then do;
sign='+';sig='*';
end;
else do;
sign='';sig='';
end;
keep _name_ event_return ref_return dif_return sign sig;
run;
```

如表 16-47 所示的语法为标记绩效是显著高于或者低于配对样本的绩效，配对样本检定的结果如图 16-11 所示。其最后整理的结果发现，事件公司的长期买进持有报酬率皆显著低于配对样本的买进持有报酬率，表示有负的显著异常报酬率。

NAME	event_return	ref_return	dif_return	sign	sig
hold5	-2.032	-3.929	1.897	+	**
hold6	-0.739	-2.958	2.219	+	**
hold7	0.088	-1.89	1.978		
hold8	1.278	-0.631	1.909		
hold9	0.536	0.857	-0.321		
hold10	0.583	-0.627	1.21		
hold11	2.102	1.539	0.563		
hold12	3.224	4.811	-1.587		
hold13	5.489	6.671	-1.182		
hold14	4.244	2.958	1.286		
hold15	6.236	2.963	3.273	+	***
hold16	4.617	1.148	3.469	+	**
hold17	9.701	4.385	5.316	+	***
hold18	9.205	5.434	3.771	+	**

图 16-11

综合日历期间投资组合、配对投资组合以及配对样本的异常报酬率检定，结果显示：公司更换财务主管在长期会有负的异常报酬率，但是本程序的范例并非是严格控制配对样本的结果，如果做进一步的产业控制以及动能绩效的控制，其结果或许会有所不同。

近来亦有不少研究认为检定市场绩效是不足的，还要求检验事件公司与配对公司的会计绩效，如资产报酬率或者股东权益报酬率等，其程序撰写逻辑亦如本节的范例，先根据配对的准则使用 `proc sql` 挑选出合适的配对公司，接着采用随机抽样的方式抽取出一家配对公司（不需要抽取 1000 次），最后再检定样本公司与配对公司的会计绩效即可，将本节的程序语法做适当的修改，应该就可以完成会计绩效的事件研究法检定，在此不再重复说明。

16.5 总结

事件研究法是财务研究中经常使用的研究方法，本章介绍事件研究法的长短期异常绩效的程序撰写，实际上异常报酬率往往是初步的探讨，近来许多研究亦使用异常报酬率作为被解释变量，进而去分析探讨影响异常报酬率的可能原因，本章节所使用的语法较为繁复，读者在阅读使用上需要更加仔细。

第 17 章

特殊议题

在财务研究工作中大都依赖统计假设与统计检定，是先对母体特性做一个适当的描述或假设，然后利用抽取出来的随机样本来检定该推论的正确性，本章在 17.1 节先介绍均值抽样分配的程序语法，读者在不确定统计量的情况下，可撰写抽样分配的语法，以求得近似的统计量做检验。

而在实际市场的数据上，研究者所面临的数据未必会符合正态性或者独立性的假设，甚至其真实的母群体分配都未知，此时便需要使用拔靴法来画出实际分配以判定显著性，故本章在 17.2 节使用简单的回归来介绍初步拔靴法的程序语法。

而在财务研究上，常需要使用 3 因子以及 4 因子模型甚至是 5 因子模型的报酬率来进行分析。因此，我们在 17.3 节，根据 Fama and French (2016) 以及 Carhart (1997) 提出的 5 因子模型以及动能因子，同时参照 French 网页上的信息，构建求 5 因子报酬率所需的数据¹。

1 http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html 记载了 SMB 以及 HML 的构建方式，并且也介绍了其动能因子的做法，以及求 5 因子报酬率所需的数据。

17.1 均值抽样分配

在先前的数据分析中有提到 t 统计量，以及其显著水平，本节介绍如何在 SAS 中执行均值抽样分配的数据仿真。在研究中，比如有 10000 笔数据就是所谓的母群体。而在一般的统计教科书中通常提到，当我们从一个正态母群体中重复进行随机抽样 M 次，每次抽出 N 个样本，其每次抽样的平均为 X_1, X_2 的均值为 μ ，其均值的抽样分配会是正态分配以及 t 分配。

实际上未必需要知道均值的抽样分配是否为 t 分配或者正态分配，如果能够知道该抽样分配的 cut point 的数据，就能够分辨其是否有显著，接下来以 example_17_1.sas 为例来介绍均值抽样分配的语法，产生 10000 笔数据的语法如表 17-1 所示。

表 17-1

<pre>data a; do i=1 to 10000; a=10+3*rannor(1); output; end; keep a; run; proc means data=a mean var; var a; run;</pre>

在如表 17-1 所示的程序中，先产生 1 组均值为 10、方差为 9 的 10000 笔数据的数据。求得抽样 M 次抽取 N 样本的平均值抽样分配的语法如表 17-2 所示。

表 17-2

<pre>%macro t(m,n); proc surveyselect data=a out=b method=srs rep=&m n=&n seed=1; run; proc means data=b noprint; var a;by replicate; output out=b (drop=_freq_) mean=a; run; proc means data=b noprint; var a; output out=c(drop=_freq_) mean=a1 std=a2;</pre>

续表

```
run;
data b;
merge b c;by _type_;
t=(a-a1)/a2;
M=&m;
N=&n;
run;
proc univariate data=b noprint ;
var t;
by M n;
output out=point pctlpts  =0.5 1 2.5 5 10 90 95 97.5 99 99.5
                        pctlpre  =p;
run;
proc append base=tdis data=point;
run;
quit;
%mend t;
```

表 17-2 是一个宏程序，利用了 proc surveyselect、proc means 以及 proc univariate 三种程序语法。

第一步要求 SAS 从 a 中抽取 m 次，每次抽取 n 个样本；第二步，计算这 m 次中，n 个样本的均值；第三步，根据第二步得到了 m 次的均值，再以该均值样本求算出 m 次抽样均值以及标准偏差，并与第二步的结果合并；第四步，计算每次抽样样本的 t 分数。

最后利用 proc univariate 的程序语法，可以求得抽样 m 次样本 t 值的分布，要求输出显著水平百分位数的门限时，考虑双尾显著性，需输出 0.5, 2.5, 5, 95, 97.5, 99.5 的百分位数值值的 t 值，考量单尾显著性，则需输出 1, 5, 10, 90, 95, 99 的百分位数值值的 t 值，这样一来，使用者就可以得到显著水平的 t 值对照表，最后的对照，如图 17-1 所示。

构建均值抽样分配检定的语法如表 17-3 所示。

表 17-3

```
proc datasets;
delete tdis;
run;
%ot(100,6);
%ot(150,6);
%ot(200,6);
%ot(500,6);
%ot(1000,6);
%ot(100,10);
%ot(150,10);
```

续表

		%t(200,10);									
		%t(500,10);									
		%t(1000,10);									
		%t(100,30);									
		%t(150,30);									
		%t(200,30);									
		%t(500,30);									
		%t(1000,30);									
		%t(100,100);									
		%t(150,100);									
		%t(200,100);									
		%t(500,100);									
		%t(1000,100);									

由表 17-3 可见，为了重复进行程序，先使用 proc datasets 将 tdis 这个文档删除，再由表 17-2 所示的宏程序重新产生该文档，最后结果如图 17-1 所示。

	M	N	p0_5	p1	p2_5	p5	p10	p50	p95	p97_5	p99	p99_5	▲
1	100	6	-2.212853137	-2.012445023	-1.785234589	-1.656193881	-1.364856721	1.268163297	1.8310595327	1.9338213701	2.0480967558	2.0531161604	
2	150	6	-2.338729185	-1.929891189	-1.891777871	-1.765978033	-1.410181041	1.2662833305	1.7644558601	1.887669874	2.0023777297	2.0126174514	
3	200	6	-2.263775895	-2.06891615	-1.915244631	-1.769628449	-1.355688867	1.280497925	1.7483896804	1.9696790056	2.1001272168	2.1155052665	
4	500	6	-2.396043372	-2.187726658	-2.039281569	-1.674465651	-1.295210555	1.2459271618	1.593786385	1.9446981261	2.1772758143	2.2123301658	
5	1000	6	-2.661221613	-2.385165076	-2.052395093	-1.657026798	-1.302708068	1.2597494877	1.6343368199	1.9594156385	2.2074004262	2.421496983	
6	100	10	-2.241269413	-2.089826392	-1.86200052	-1.51805951	-1.315755683	1.1184520565	1.4596439034	2.3445004825	2.5373138324	2.6544761738	
7	150	10	-2.189864246	-2.137955319	-1.893970621	-1.519050289	-1.283441122	1.2787243529	1.5555628726	2.1707103126	2.272743291	2.4654638968	
8	200	10	-2.178990027	-2.03459116	-1.816345967	-1.515444453	-1.282444306	1.273034989	1.7290576107	2.1315942539	2.2544407622	2.3661534659	
9	500	10	-2.19790154	-1.986819278	-1.82198066	-1.563163927	-1.288171977	1.2139411684	1.7081249787	2.0317341106	2.3283649461	2.5240634256	
10	1000	10	-2.165986857	-1.993030922	-1.852388251	-1.591236692	-1.310556928	1.2628864034	1.6952232845	2.0177007941	2.3677285801	2.6420573658	
11	100	30	-2.321843361	-2.309715604	-2.073084219	-1.767543073	-1.268781061	1.2576003031	1.6782897317	1.8800876619	2.1034250766	2.1198679776	
12	150	30	-2.295719519	-2.272236825	-1.981634594	-1.737779938	-1.276209497	1.3098178406	1.6750614775	2.0044716696	2.1469860947	2.4400319274	
13	200	30	-2.282643144	-2.164606958	-1.919396555	-1.744216818	-1.26301242	1.2965697569	1.7916506895	2.0485305668	2.2378380518	2.3717310755	
14	500	30	-2.3706947	-2.271991296	-1.832433117	-1.62115704	-1.272692803	1.2598437498	1.7709104307	1.9885701581	2.3354965749	2.9482920727	
15	1000	30	-2.339071381	-2.23215431	-1.932995741	-1.640797743	-1.360055917	1.2676095936	1.6695327543	1.9241988003	2.3134897866	2.788117488	
16	100	100	-3.024914966	-2.621133149	-1.935330445	-1.608827755	-1.237417854	1.1862972647	1.4547198966	1.6902920867	2.5054091258	2.5632620206	
17	150	100	-2.802543015	-2.760958014	-2.071336449	-1.573928237	-1.265044142	1.2535316938	1.4668299718	2.0191225456	2.2572585178	2.4437754367	
18	200	100	-2.712403457	-2.606805203	-1.991742212	-1.70469293	-1.345600446	1.1490303509	1.4799402066	1.6657705345	2.2499778726	2.3949109913	
19	500	100	-2.729998036	-2.418260513	-1.831012734	-1.63843547	-1.326958837	1.2491948383	1.5920373673	2.0270896482	2.4176580164	2.4805618766	
20	1000	100	-2.567749246	-2.243212946	-1.945813849	-1.678113437	-1.349483731	1.2276830773	1.5987844892	1.9401998021	2.3688889078	2.440412118	

图 17-1

由图 17-1 所示的结果读者可以确定均值是否达到显著水平了，在蒙特卡罗模拟法中，常常需要仿真数据上万次，就可以用这种方法来找出显著水平，接下来我们介绍拔靴法。

17.2 拔靴法 (Bootstrap method)

许多传统的统计分析方法是基于正态性的假定之下，包括相关分析、回归分析、t 统计量以及方差分析，当正态性的假定违反时，这些统计方法就会失灵。拔靴法是以数据为基础 (data-based) 的模拟方法，拔靴法最适用于当样本数目有限，以重复抽样原有样本的方式，来求得较精确的抽样分配。而在执行上常常需要借助于计算机，所以随着计算机运算能力的提高，拔靴法的应用也就越来越广泛。

拔靴法是由 Efron (1979) 提出的统计方法，其概念是利用样本以重复取出、放回的方式仿真出一个随机样本，再由仿真出来的随机样本的统计量进行估计与检定。其思路是利用样本数据重复抽取，以模拟出母体的分配，再由模拟出来的母体特征进行估计与检定，因此并不需要知道母体的实际分配状况，其优点在于不必假设母体的分配却可以掌握母体分配的特性，同时也适用于小样本。拔靴法的示意图如图 17-2 所示。

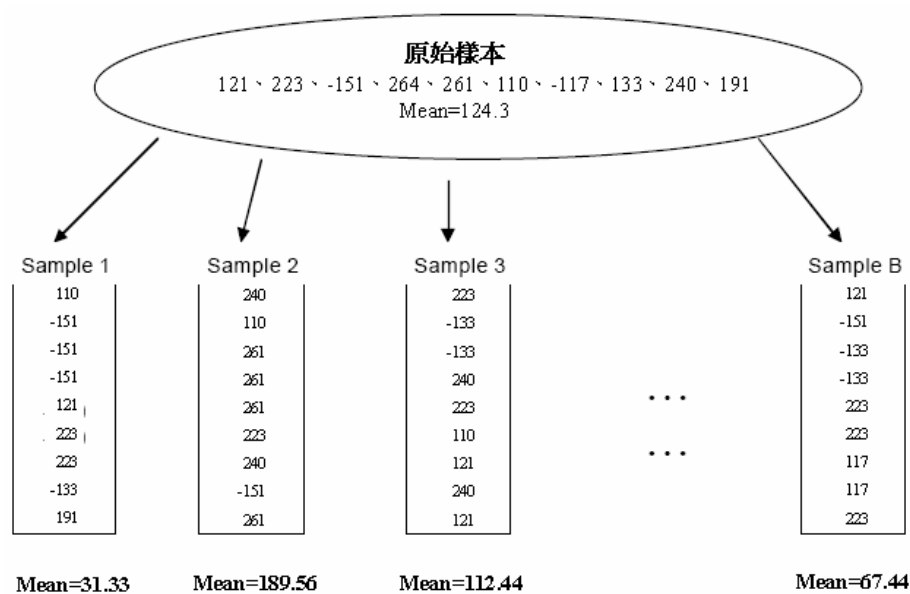


图 17-2

拔靴法的方式是随机抽样，同时置回已经抽取的样本，进行重复性的多次抽样（重复取样），来模拟真实的分配。假设 $X=(x_1, x_2, \dots, x_n)$ ，其参数估计 $\hat{\theta}(x_1, x_2, \dots, x_n)=\hat{\theta}$ ，采用取出后放回的方式从原始样本 X 内每次取足样本数为 n 的数据 $X^*=(x_1^*, x_2^*, \dots, x_n^*)$ ， X^* 称为拔靴样本，做了 B 次后，便可以得到 B 笔拔靴样本，并可估计 B 笔的 $\hat{\theta}^*(x_1^*, x_2^*, \dots, x_n^*)=\hat{\theta}^*$ ，若 $\hat{\theta}$ 为样本均值，那么便可以取得 B 笔 $\hat{\theta}^*$ ，根据弱大数法则，当拔靴的次数 B 或者抽取的 n 样本趋近无限大时， $\text{Var}(\hat{\theta}^*) \rightarrow \text{Var}(\hat{\theta})$ ，一般来说，拔靴法的功用就在于估计 $\hat{\theta}$ 的标准误差，以下以 example_17_2.sas 为例来介绍如表 17-4 所示的拔靴法的简单回归语句。

表 17-4

DM'LOG;CLEAR;OUT;CLEAR';
option nonotes nolabel;
libname aa 'D:\The Application of SAS in Financial Research\SAS data\four factor';
libname bb 'D:\The Application of SAS in Financial Research\SAS data\monthly price';
data a;
set bb.price;

续表

```

mv=log(mv);
mv2=mv**2;
turn=turn*0.01;
turn2=turn**2;
mv_turn=mv*turn;
run;
proc sort data=a;by y m;
run;
proc rank groups=2 out=a;
var ret;
ranks retr;by y m;
run;
proc sort;by y m retr;
proc means noprint data=a;
var ret mv mv2 turn turn2 mv_turn;
by y m retr;
output out=a mean= ret mv mv2 turn turn2 mv_turn;
run;
data a;
set a;
if retr=. then delete;
drop _type__freq_;
run;
proc reg data=a;
model ret=mv turn mv2 turn2 mv_turn ;
model ret=mv ;
model ret=turn;
model ret=mv turn;
model ret=mv turn mv2;
model ret=mv turn turn2;
model ret=mv turn mv_turn;
model ret=mv turn mv2 turn2 mv_turn ;
ods output ParameterEstimates=reg;
quit;

```

由表 17-4 所示的程序可见，将数据整理成每个月有两笔数据，分别是当月赢家与当月输家，可将原始样本数据由 10 万笔减少为 672 笔，方便进行之后拔靴法的模拟，其中 ods 的 reg 表格为以后要处

理的表格。回归系数的结果如图 17-3 所示。

	Model	Dependent	Variable	DF	Estimate	StdErr	tValue	Probt
1	MODEL1	ret	Intercept	1	17.29130	38.21698	0.45	0.6511
2	MODEL1	ret	mv	1	-4.66124	9.83518	-0.47	0.6357
3	MODEL1	ret	turn	1	52.00396	32.59366	1.60	0.1111
4	MODEL1	ret	mv2	1	0.16626	0.61930	0.27	0.7884
5	MODEL1	ret	turn2	1	-16.08243	5.01846	-3.20	0.0014
6	MODEL1	ret	mv_turn	1	0.36118	4.18226	0.09	0.9312
7	MODEL2	ret	Intercept	1	-15.73177	5.37268	-2.93	0.0035
8	MODEL2	ret	mv	1	2.13094	0.64249	3.32	0.0010
9	MODEL3	ret	Intercept	1	-5.26400	0.80800	-6.51	<.0001
10	MODEL3	ret	turn	1	26.95644	2.36578	11.39	<.0001
11	MODEL4	ret	Intercept	1	5.55518	5.30857	1.05	0.2957

图 17-3

虽然该表格有相关的系数值以及标准误差的数据，但是在此处假设标准误差并非是有误差的，需要进行拔靴法才能估计出需要的标准误差，并重新计算 tValue。

如表 17-5 所示的进行 10000 次样本抽样的程序就是执行拔靴法抽取的指令，和先前进行均值抽样分配的程序一样，但是在此处增加使用的 method 是 urs，这是因为拔靴法采用的是置回抽样的方法。该指令要求 SAS 抽出和原有样本数目相同的样本，抽取 10000 次以及设定随机数的起始值为 1。

表 17-5

proc surveyselect data=a out=boot method=urs rep=10000 rate=1 seed=1; run;
--

求算 10000 次回归系数值的程序如表 17-6 所示。

表 17-6

proc reg data=boot noprint outest=b; model ret=mv turn mv2 turn2 mv_turn; model ret=mv; model ret=turn; model ret=mv turn; model ret=mv turn mv2; model ret=mv turn turn2; model ret=mv turn mv_turn; model ret=mv turn mv2 turn2 mv_turn; by replicate; freq numberhits; run; proc sort data=b;by _model_; run; proc means noprint data=b; var intercept mv turn mv2 turn2 mv_turn;

续表

```
by _model_ ;  
output out=b1 std= Intercept mv turn mv2 turn2 mv_turn ;  
run;  
%let bit=3;
```

如表 17-6 所示的程序便是将样本依照 replicate 进行回归，所谓 replicate 就是第 i 次的抽取样本，而在此处，进行回归时使用了 freq numberhits 这个指令，freq 是指出现的频率，是告诉 SAS 该观测值总共出现几次。抽样回归的结果如图 17-4 所示。

	Replicate	y	m	ret	ret	mv	mv2	turn	turn2	mv_turn	Numberhits
1	1	1980	2	0	1.20125	7.3770747436	55.569466504	0.0583125	0.0078777042	0.4126625745	1
2	1	1980	2	1	11.14	6.8603504827	48.082840326	0.1891342105	0.0746390982	1.2108739438	1
3	1	1980	3	1	2.0365853659	7.2596451167	53.904169805	0.10759	0.031254391	0.7219596236	1
4	1	1980	4	0	-4.65075	7.0537502195	50.83571377	0.0729578947	0.0114348184	0.4935396247	3
5	1	1980	5	1	3.792	7.1992153541	53.054410141	0.0863275	0.0171227423	0.5676785745	2
6	1	1980	6	0	-11.05461538	6.6648293875	45.241940398	0.0756945946	0.0110051711	0.4723602014	1
7	1	1980	7	0	4.491025641	6.8602242814	48.672947395	0.1180594595	0.042022407	0.7543654361	1

图 17-4

如图 17-4 所示，第 4 笔数据在抽样时被抽出了 3 次，第 5 笔数据被抽出了 2 次，不能将之看作 1 笔观测值，而是总共产生了 5 个观测值，当取得的样本数据为次数分配的形式表示时，在进行回归时，亦可使用 freq 的指令。

接下来便依照模型计算每个变量的标准误差，该过程就如同先前所介绍的回归方法，也就是 Fama-MacBeth 的做法，但是此处是以这 10000 次的系数标准误差作为真实样本的系数标准误差的，接着将原始模型的系数值除以该标准偏差，便可以得到修正误差的 t 值。

重新计算回归系数 t 值的语法如表 17-7 所示。抽样分配回归系数的 t 值结果如图 17-5 所示。

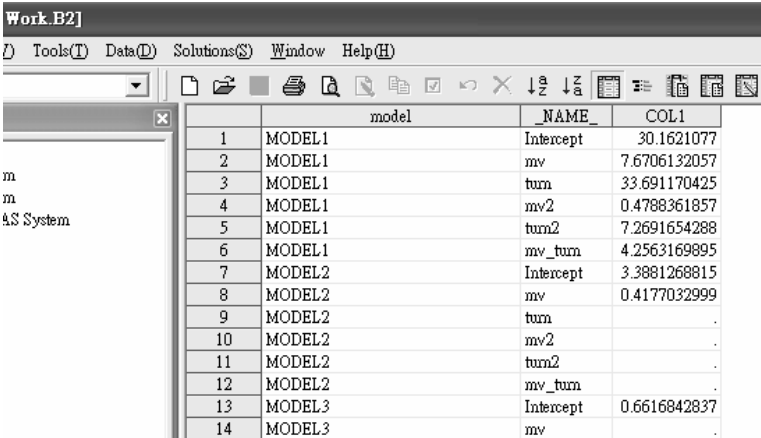
表 17-7

```
data b1;  
set b1;  
model=_model_ ;  
drop _type_ _freq_ _model_ ;  
run;  
proc transpose data=b1 out=b2 ;  
by model;  
run;
```

	Intercept	mv	turn	mv2	turn2	mv_turn	model
1	30.1621077	7.6706132057	33.691170425	0.4788361857	7.2691654288	4.2563169895	MODEL1
2	3.3881268815	0.4177032999	MODEL2
3	0.6616842837	.	2.8777426706	.	.	.	MODEL3
4	3.4362571231	0.4586603371	3.2912305706	.	.	.	MODEL4
5	30.945114617	7.9436476819	3.3539981464	0.4956593381	.	.	MODEL5
6	3.4990806575	0.4707659218	8.6292188552	.	5.9368379567	.	MODEL6
7	6.3669766972	0.7485266399	29.856851485	.	.	3.2803865504	MODEL7
8	30.1621077	7.6706132057	33.691170425	0.4788361857	7.2691654288	4.2563169895	MODEL8

图 17-5

SAS 的 table 为每个 model 所有的回归系数值的标准偏差，经由转置后变成 b2 的形式，如图 17-6 所示。系数值与 t 值的整理语法如表 17-8 所示。



	model	_NAME_	COL1	
1	MODEL1	Intercept	30.1621077	
2	MODEL1	mv	7.6706132057	
3	MODEL1	turn	33.691170425	
4	MODEL1	mv2	0.4788361857	
5	MODEL1	turn2	7.2691654288	
6	MODEL1	mv_turn	4.2563169895	
7	MODEL2	Intercept	3.3881268815	
8	MODEL2	mv	0.4177032999	
9	MODEL2	turn	.	
10	MODEL2	mv2	.	
11	MODEL2	turn2	.	
12	MODEL2	mv_turn	.	
13	MODEL3	Intercept	0.6616842837	
14	MODEL3	mv	.	

图 17-6

表 17-8

```
data a;
set reg;
keep model variable estimate;
run;
data b;
set b2;
variable=_name_;
if col1=, then delete;
keep model variable col1;
run;
data a;
merge a b;
est=round(estimate,0.001);
tvalue=round(estimate/col1,0.01);
if abs(tvalue)>=2.58 then sig='***';
if 2.58>abs(tvalue)>=1.96 then sig='**';
if 1.96>abs(tvalue)>=1.65 then sig='*';
a=')';
esti=est||sig;
tv='('||tvalue||a;
keep model variable esti tv;
```

续表

```

run;
data t;
set a;
retain t 0;
t=t+1;
if model='MODEL1' then output;
keep variable t;
run;
proc sort;by variable;
run;

```

系数值与 t 值的整理结果如图 17-7 所示。在实证研究中希望呈现的是一个一个的模型，这可以通过转置来完成这个目标，使用转置语法就可以完成这个目标，基本上需要多方尝试才有可能转成希望的形态，所以要用如表 17-9 所示的系数值转置语法进行转置。

Model	Variable	esti	tv
MODEL1	Intercept	17.291	(0.45)
MODEL2	Intercept	-15.732***	(-3.54)
MODEL3	Intercept	-5.264***	(-6.1)
MODEL4	Intercept	5.555	(1.24)
MODEL5	Intercept	9.603	(0.24)
MODEL6	Intercept	5.82	(1.29)
MODEL7	Intercept	-7.394	(-0.89)
MODEL8	Intercept	17.291	(0.45)
MODEL1	mv	-4.661	(-0.47)
MODEL2	mv	2.131***	(3.9)
MODEL4	mv	-1.385**	(-2.33)
MODEL5	mv	-2.411	(-0.24)
MODEL6	mv	-1.846***	(-3.03)
MODEL7	mv	0.133	(0.14)
MODEL8	mv	-4.661	(-0.47)
MODEL1	turn	52.004	(1.21)
MODEL3	turn	26.956***	(7.25)
MODEL4	turn	29.579***	(6.95)

图 17-7

表 17-9

```

proc transpose data=a out=boot_ols;
var esti tv;
id model;
by t variable;
run;

```

系数值转置后的结果如图 17-8 所示,但是仍需要经过一些微调才适合放置在表格上,其中 t_ _name_

以及 MODEL1 都需要删除，而 t 值不需要表示其 Variable。采用拔靴抽样法的回归结果输出语法如表 17-10 所示。

	t	Variable	_NAME_	MODEL1	MODEL2	MODEL3	MODEL4	MODEL5
1		1 Intercept	esti	17.291	-15.732***	-5.264***	5.555	9.603
2		1 Intercept	tv	(0.45)	(-3.54)	(-6.1)	(1.24)	(0.24)
3		2 mv	esti	-4.661	2.131***		-1.385**	-2.411
4		2 mv	tv	(-0.47)	(3.9)		(-2.33)	(-0.24)
5		3 turn	esti	52.004		26.956***	29.579***	29.561***
6		3 turn	tv	(1.21)		(7.25)	(6.95)	(6.84)
7		4 mv2	esti	0.166				0.063
8		4 mv2	tv	(0.27)				(0.1)
9		5 turn2	esti	-16.082*				
10		5 turn2	tv	(-1.88)				
11		6 mv_turn	esti	0.361				
12		6 mv_turn	tv	(0.07)				

图 17-8

表 17-10

```

data boot_ols;
set boot_ols;
if _name_='tv' then variable="";
drop t type _name_ model1;
run;
proc export data=boot_ols
outfile='D:\The Application of SAS in Financial Research\CH17\output\boot'
dbms=xlsw
replace;
run;

```

运行如表 17-10 所示程序之后直接将之输出直式回归模型的结果，在程序的最后，一样可以自行修改所想要的路径以及文件名。ols 的简单拔靴法结果如表 17-11 所示。

表 17-11

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Intercept	-15.732***	-5.264***	5.555	9.603	5.82	-7.394	17.291
	(-3.54)	(-6.1)	(1.24)	(0.24)	(1.29)	(-0.89)	(0.45)
mv	2.131***		-1.385**	-2.411	-1.846***	0.133	-4.661
	(3.9)		(-2.33)	(-0.24)	(-3.03)	(0.14)	(-0.47)
turn		26.956***	29.579***	29.561***	54.688***	84.749**	52.004
		(7.25)	(6.95)	(6.84)	(5.23)	(2.2)	(1.21)
mv2				0.063			0.166
				(0.1)			(0.27)
turn2					-15.766**		-16.082*

续表

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
					(-2.25)		(-1.88)
mv_turn						-6.344	0.361
						(-1.5)	(0.07)

在本节中，仅仅介绍了拔靴法在回归分析上的运用，实际上拔靴法还可用在均值检定、相关系数分析等。均值检定，读者可参考本书检定买进持有异常报酬率的配对样本的做法，相关系数则以 Kumar and Lee (2006) 关于检验投资者买卖行为是否有显著相关的做法，在进行拔靴法时，只要掌握 `proc surveysselect` 的语法¹就能够顺利进行。

17.3 构建 5 因子与动能因子报酬率

自从 Fama and French (1993) 提出 3 因子模型以来，Carhart (1997) 延伸了 3 因子模型，加入了动能因子来解释共同基金的绩效持续性，在投资学的研究领域，3 因子模型与 4 因子模型已经变成不可或缺的研究方法，而 Fama and French (2016) 更进一步提出了 5 因子定价模型，并在 2015 年将 5 因子报酬率上线提供研究者使用。在进行美国的实证时，研究者仅需要在 French 的网页上便能够取得相关的报酬率数据，然而，若使用其他地区数据的话，便需要自行建构，以下我们便逐步撰写建构 5 因子中的 SMB、HML、RMW、CMA 以及动能因子的报酬率数据。

Fama and French (1992; 1993) 利用市值以及账市比 (BM) 将股票形成 6 个投资组合，其中在 t 年 6 月底时，利用当时的股票市值将 NYSE 的股票依照中位数分为大 (B)、小 (S) 两组，接着再将 Amex 以及 NASDAQ 的股票归类依照 NYSE 的中位数，将股票归类为大或者小型股的投资组合，同时将股票依照 t-1 年 12 月底时的账市比将股票分为高 (H) BM (前 30%)、中 (M) BM (30%到 70%) 以及低 (L) BM (后 30%) 的股票，其中账面价值的计算方式为，股东权益的账面价值扣除特别股价值，并且加回递延所得税与递延贷项；同时如果所求得的账面价值小于 0，则不列入计算，如此将股票取其交集，求得 SH、SM、SL、BH、BM、BL 6 个投资组合，最后计算 t 年 7 月到 t+1 年 6 月各个投资组合的市值加权平均报酬率，最后，可求得 SMB 以及 HML 的报酬率数据，其中 $SMB=1/3(SH+SM+SL-BH-BM-BL)$ 、 $HML=1/2(SH+BH-SL-BL)$ ，最后为了避免 IPO 的效应，构建投资组合时，如果股票上市未满 2 年，便会将其排除。

而 Fama and French (2016) 更进一步构建出了 RMW 以及 CMA 因子，这两个因子的构建采用与 HML 因子相同的方式，分别与规模交会之后构建出 2 乘 3 的投资组合，其中 RMW 是采用净利润率 (operating profitability; OP) 构建出稳健 (Robust) 以及衰落 (Weak) 投资组合；而 CMA 则采用投资 (investment) 来构建出保守 (conservative) 和积极 (aggressive) 投资组合，此处特别说明，虽然 CMA

1 SAS 亦提供了 `surveymeans`、`surveylogistic` 以及 `surveyreg` 的语法进行拔靴估计，有兴趣的读者可以做进一步的研究，否则我们推荐使用 `proc surveysselect`。

为一个投资因子，但在 Fama and French (2016) 的论文中，该因子的构建是采用总资产的成长率，而不是资本支出这个常用的投资变量。

综合上述，可以一一列出构建 SMB、HML、RMW 以及 CWA 因子的条件，这是撰写程序的第一个步骤，然后写下执行条件和步骤，如此在撰写程序时才不会不知从何下手，构建 SMB 与 HML 的具体步骤如下。

第 1 步，删除 t 年 12 月底时上市未满 24 个月的股票。

第 2 步，删除 t 年 6 月底时，没有市值数据的股票。

第 3 步，计算 t-1 年 12 月底时的账面价值、资产成长率以及净利润率的数据。

第 4 步，删除 t-1 年 12 月底时账面价值小于 0 的公司，并计算其账市比。

第 5 步，合并 t 年 6 月底的有市值数据与 t-1 年 12 月底的账市比、成长率以及净利润率。

第 6 步，将有完整数据的公司根据市值分为 2 个投资组合 (S、B)、根据账市比分为 3 个投资组合 (HBM、MBM、LBM)、根据净利润率将公司分为 3 个投资组合 (ROP、MOP、WOP)、根据成长率分为 3 个投资组合 (CInv、MInv、AInv)。

第 7 步，将其与 t 年 7 月到 t+1 年 6 月的股票报酬率合并，计算其每一期的市值加权平均报酬率¹。

以下，以 example_17_3.sas 为例来构建 SMB、HML、RMW、CMA 的数据。

样本报酬率整理的语法如表 17-12 所示。

表 17-12

<pre>option nonotes nolabel; libname aa 'D:\The Application of SAS in Financial Research\CH17\data'; data a; set aa.month_return; run; proc sort data=a;by stkcd year month; run; /*至少需要 2 年数据*/ data a; set a;by stkcd; retain age 0; age=age+1; if first.stkcd then age=1; keep year month stkcd mv ret age; run;</pre>

在如表 17-12 所示的程序中，先将数据依照公司代码、月份排序，接着对公司进行年龄编码，将公司最早出现于数据中编码为 1，接着将其累加，此处需特别注意，这是在无法取得公司正确的上市

1 此处合并的股票报酬率数据的频率可以是月数据，也可以是日数据。

日期时所采用的权衡方法，利用最早被纳入数据库的日期为其起始日，因此在构建样本时，会损失取得前 2 个年度的数据。

处理投资与账面价值数据的语法如表 17-13 所示。

表 17-13

<pre>data mv; set a; if month=6 then output; keep year stkcd mv; run; proc sort data=mv;by year stkcd; run; /*Fama and French 1993 require the compustat data must exist two year*/ proc sort data=aa.finance out=fin;by stkcd year; run; data fin; set fin; Inv=dif(asset)/lag(asset); book=sum(equity,-prefer_stk); if stkcd^=lag(stkcd) then inv=.; if dif(year)^=1 then inv=.; if book>0 and inv^=, and op^=;</pre>
<pre>run;</pre>

在如表 17-13 所示的程序中，取出 t 年 6 月底的市值数据以及 t-1 年 6 月底的市值数据，根据 Fama and French (1993) 要求 compustat 的数据存在 2 年，因此要求 12 月底的数据时，公司需要满足上市两年的条件。

合并所需分组数据的语法如表 17-14 所示。

表 17-14

<pre>proc sql; create table finance as select a.stkcd, a.year+1 as year,b.book/a.mv as BM,b.inv,b.op /* 采用 t-1 年的财报数据与 t 年的市值数据合并 */ from a, fin b where a.stkcd=b.stkcd and a.month=b.month and a.year=b.year and a.age>=24; create table final as select</pre>

续表

```
a.stkcd, a.year, a.mv,b.bm, b.op,b.inv
from mv a,finance b
where a.stkcd=b.stkcd and a.year=b.year
order by a.year;
quit;
```

由表 17-14 所示的语法便能够得到所有在 t 年 6 月底满足上市满 2 年、有市值、BM、资产成长率以及净利润率等数据，并且同时账面价值大于 0 的全部公司，接下来便将公司分出六个投资组合，此时提供一个小技巧，由于规模只分成两组，而 BM、资产成长率以及净利润率则分成 30%、40%、30%，在处理上，可以先将股票都分成 10 组，之后再用条件句的语法将其分组即可。

进行股票样本分组的语法如表 17-15 所示。

表 17-15

```
proc rank data=final out=final groups=10;
var mv bm op inv;
by year;
ranks mvr bmr opr invr;
run;
data final;
set final;
bmr=bmr+1;
mvr=mvr+1;
opr=opr+1;
invr=invr+1;
if bmr<=3 then b='L';
else if bmr<=7 then b='M';
else b='H';
if opr<=3 then R='W';
else if opr <=7 then R='M';
else R='R';
if invr<=3 then C='C';
else if invr<=7 then C='M';
else C='A';
if mvr<=5 then s='S';
else s='B';
SB=s||b;
SC=s||c;
SR=s||r;
keep year stkcd SB SC SR;
run;
```


在如表 17-15 所示的程序中，利用条件句的语法以后，再将投资组合名称合并，就完成了数据的分组工作，接下来便要针对数据合并的工作做处理，因为 t 年 6 月底分组的数据是为了与 t 年 7 月到 $t+1$ 年 6 月的报酬率数据进行合并。将分组数据处理为 12 个月份的语句如表 17-16 所示。

表 17-16

```

data final;
set final;
do month=1 to 12;
output;
end;
run;
data final;
set final;
if month<=6 then year=year+1;
run;
proc sort data=final;by stkcd year month;
run;

```

在如表 17-16 所示的程序处理完毕之后，就可以针对报酬率的数据做合并了，下面先针对股票报酬率进行处理，在此之前，先处理动能组合，以便同时计算动能因子的数据。有关动能因子的数据，根据 Carhart (1997) 提出的内容，其构建方式是采用前 2 个月到前 12 个月的报酬率数据分组，将股票分成最高 30% 的股票以及最低 30% 的股票，并且计算其简单平均报酬率的差额，按该方式来处理动能投资组合的数据。求得动能报酬率数值与分组的语法如表 17-17 所示。

表 17-17

```

%macro mom;
data mom;
set a;
ret=1+0.01*ret;
%do i=2 %to 12;
r&i=lag&i(ret);
%end;
pr=geomean(of r2-r12);
if nmiss(of r2-r12)>0 then pr=.;
if code^=lag12(code) then delete;
run;
proc sort data=mom;by year month;
run;
proc rank data=mom out=mom groups=10;

```

续表

```

var pr;
ranks prr;
by year month;
run;
proc sort data=mom;by stked year month;
run;
data mom;
set mom;by stked;
prr=lag(prr);
if first.stked then prr=.;
prr=prr+1;
if prr<=3 then p='L';
if prr>=8 then p='W';
if p="" then delete;
keep year month stked p;
run;
%mend;
%mom;

```

由于需要计算前 2 个月到前 12 个月的报酬率,因此需要使用宏语句来运算,使用 mom 作为命名,最后取得 mom 这个文档,接下来撰写宏语句来计算因子报酬率的数据,其中 final 与 mom 分别是 SMB、HML、RMW、CMA 以及 MOM 所需要的文档,不需做另外命名。构建因子报酬率的语法如表 17-18 所示。

表 17-18

```

%macro factor(firm,time,weight,in,out);
proc sort data=&in out=ss;by &firm &time;
run;
data ss;
set ss;by &firm;
&weight=lag(&weight);
if first.&firm then delete;
run;
data ss;
merge ss final;by &firm year month;
run;
proc sort data=ss;by &time sb;
run;
proc means noprint data=ss(where=(sb^="));

```

续表

```

var ret;
weight &weight;
by &time sb;
output out=sb(drop=_type__freq_) mean=ret;
run;
proc transpose data=sb out=sb;
var ret;
id sb;
by &time;
run;

data sb;
set sb;

/*revise 20150626*/
SMB_BM=1/3*(sh+sm+sl-bh-bm-bl);
HML=1/2*(sh+bh-sl-bl);
keep &time SMB_BM hml;
if SMB_BM=. then delete;
/*revise 20150626*/

run;
proc sort data=ss;by &time sr;
run;
proc means noprint data=ss(where=(sr^="));
var ret;
weight &weight;
by &time sr;
output out=sr(drop=_type__freq_) mean=ret;
run;
proc transpose data=sr out=sr;
var ret;
id sr;
by &time;
run;
data sr;
set sr;

/*revise 20150626*/
SMB_OP=1/3*(sR+sm+sW-bR-bm-bW);

```

```

        RMW=1/2*(sr+br-sw-bw);
        keep &time SMB_OP RMW;
        if RMW=. then delete;
        /*revise 20150626*/

run;
proc sort data=ss;by &time sc;
run;
proc means noprint data=ss(where=(sc^=""));
var ret;
weight &weight;
by &time sc;
output out=sc(drop=_type__freq_) mean=ret;
run;
proc transpose data=sc out=sc;
var ret;
id sc;
by &time;
run;
data sc;
set sc;

        /*revise 20150626*/
        SMB_INV=1/3*(sC+sm+sA-bC-bm-bA);
        CMA=1/2*(sc+bc-sa-ba);
        keep &time SMB_INV CMA;
        if CMA=. then delete;
        /*revise 20150626*/

run;
proc sort data=&in out=ss;by &firm &time;
run;
data ss;
merge ss mom;by &firm year month;
run;
proc sort data=ss;by &time p;
run;
proc means noprint data=ss;
var ret;
by &time p;

```

续表

```

output out=carhart mean=ret;
run;
proc transpose data=carhart out=carhart;
var ret;
id p;
by &time;
run;
data carhart;
set carhart;
MOM=W-L;
keep &time MOM;
run;
data &out;

    /*revise 20150626*/
    SMB=(SMB_bm+SMB_op+SMB_INV)/3;
    if nmiss(smb,hml,rmw,cma,mom)=0 then output;
    DROP SMB_BM SMB_OP SMB_INV;
    /*revise 20150626*/

run;
%mend;
%factor(stkcd,year month,mv, a, six_month);
/*add 20150626*/
proc sort data=aa.mrmrf;by year month;
run;
data six_month;
merge aa.mrmrf six_month;by year month;
if hml=. then delete;
run;
/*add 20150626*/

```

在如表 17-18 所示的宏语句中，指定了 5 个变量，firm、time、weight，in 以及 out，firm 是公司变量，在此处是 stkcd，time 为时间变量，可以指定为月份或者日，weight 为计算市值加权平均报酬率的权重，我们多以 mv 命名，in 为报酬率来源文档，out 为最后因子报酬率的输出文档，接下来读取这个构建日报酬率的数据。整理出日因子报酬率文档的语法如表 17-19 所示。

表 17-19

```
data a;
set aa.daily_return;
keep stkcd year month day ret mv;
run;
%factor(stkcd,year month day,mv,a,six_daily);
/*add 20150626*/
proc sort data=aa.drmrf;by year month day;
run;
data six_daily;
merge aa.drmrf six_daily;by year month day;
if hml=. then delete;
run;
/*add 20150626*/
```

在如表 17-19 所示的程序中，与月数据不同之处在于，在时间变量里指定的是 year month day，即年、月、日 3 个变量，如此一来，就完成 SMB、HML、RMW、CMA 以及 MOM 因子报酬率的每月以及每日的数据构建了，之后读者再抓取市场报酬率以及无风险利率，就可以采用因素模型进行后续的研究分析了。

17.4 总结

本章简单介绍了抽样分配、拔靴法以及因子报酬率的构建，其中因子报酬率的构建在财务研究上有广泛的应用，有兴趣的读者，可以自行做进一步的研究与延伸。在拔靴法中，本章仅介绍简单的概念，读者需要自行针对其面临的情景采取适合的做法。

后记 SAS 在财务研究上应用的缘起

虽然祥萱老师一直夸奖我程序撰写得好，一直要我要撰写一本有关 SAS 程序的书。其实在我内心深处一直都明白，自己的程序能力不够，很多程序我其实也只是复制再贴上，知道修改哪边就可以跑出东西来，至于到底是为什么，我不清楚，只是随便地整理自己的语法而已，甚至我连 SAS 中可以看 help 来查语法都不会。

像我这种状况，请不要认为我的程序功力很厉害，但是，祥萱老师仍然深信不疑。

有一天，老师希望我研究一下 Heckman two stage 怎么跑。虽然，我们从来没用它跑过实证，但是，他那随口的一问，Heckman two stage 便开启了我的 SAS 之路。

我从网上找到了一个语法，这个语法让我认识了如何读外部的 SAS 程序以及 %macro 真正的意义，两者合而为一，就是标准化的程序，也就是套装语法。

我开始专研一些常用的语法，撰写一些套装化的程序，其中时间花得最多的还是回归语法，这是一个再简单不过的程序，其他人在跑回归的时候，程序不会超过 10 行，我写的程序则会超过 100 行，但是，却有着绝佳的妙处。

后来老师希望可以搜寻文字的东西（新闻内容），我找到了 SQL 语法，这个语法，开启了我学习 SAS 的另外一个境界，因为这个语法，我认为自己写出了世界上最快的事件研究的语法。

因为祥萱老师相信我可以学会 Heckman two stage 的语法，也因为祥萱老师相信我可以用 SAS 来搜寻文字，所以我撰写程序的能力这两年来在跳跃式地成长。

因此我开始撰写这本书，目的是为老师以及将来的学弟学妹留下一点东西，可以帮助他们在撰写程序上少走一点弯路。

将这本书献给献给我的祥萱老师，因为是您对我的信任，造就了这本书的诞生。

林煜恩

2010 年 3 月 31 日 于台湾东华大学

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：(010) 88254396；(010) 88258888

传 真：(010) 88254397

E-mail: dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱

电子工业出版社总编办公室

邮 编：100036